Solving the Capacitated Vehicle Routing Problem with a Genetic Algorithm

Satisfy the deterministic demand of customers from a single depot, such that the total cost is minimised and the capacity and distance restrictions are satisfied.

Bachelor Thesis Econometrics and Operational Research

Martine M.C. Verhave

June 2014

Solving the Capacitated Vehicle Routing Problem with a Genetic Algorithm

Satisfy the deterministic demand of customers from a single depot, such that the total cost is minimised and the capacity and distance restrictions are satisfied.

Author:	Maria Catharina Verhave
Student Number:	358989
E-mail:	358989mv@student.eur.nl or martje_verhave1@hotmail.com

Content:	Bachelor thesis Econometrics and Operational Research – Logistics
Study:	Bachelor Econometrics and Operational Research
Faculty:	Erasmus School of Economics
Educational institute:	Erasmus University Rotterdam
Academic year:	2013-2014
Place and time:	Rijnsburg, 29 June 2014
Supervisor:	dr. R. Spliet

Preface

The economic faculty of the Erasmus University Rotterdam, the Erasmus School of Economics (ESE), offers four different Master programmes for the study Econometrics and Operational Research. During the final phase of the Bachelor program, a bachelor thesis must be written in the field of one of the Master programmes. Due to the fact that my attention is drawn by vehicle routing, logistic planning and programming, I decided to write my thesis about a topic in Logistics. The paper that I have chosen, written by Baker and Ayechew [1], describes and analyses a Genetic Algorithm for the Capacitated Vehicle Routing Problem. I chose this subject mainly because the Capacitated Vehicle Routing Problem has a clear real-life application and it is an important topic in Logistics.

The purpose of this study is to implement the Genetic Algorithm explained in detail by Baker and Ayechew [1] and to verify the results described in the paper. Two problem instances, for which the exact solutions are known, are used to evaluate whether the algorithm is adequate or not for vehicle routing instances for which the optimal tour is known. Besides that, modifications that may improve the performance of the routing heuristic are implemented and runs are carried out to test whether or not better vehicle routes can be found.

I would like to thank my research supervisor dr. R. Spliet for the great and enthusiastic support, both during the implementation phase and the writing process. He gave generously of his time and expertise while I was writing this bachelor thesis. I would also like to thank dr. D. Huisman, the first reader of this thesis, for giving his professional evaluation.

Abstract

The Capacitated Vehicle Routing Problem involves finding vehicle routes, such that the goods that a customer demand are delivered in one visit and the travelling distance is minimised. Each customer has a known demand quantity and the vehicles in the fleet have limited capacity. The Capacitated Vehicle Routing Problem can further be extended by introducing travelling limits per vehicle, which is called the distance constrained version of the Capacitated Vehicle Routing Problem. The length of each route may not exceed this distance limit.

The general idea of a Genetic Algorithm is closely related to the theory of evolution by means of natural selection. An initial population with candidate solutions is created and this population is improved, such that the fitness of the individuals in the population becomes better through different generations. New individuals are created with crossover and small mutations are applied to these children. There must be a balance between the genetic quality and the diversity of the individuals to support efficient search. A Genetic Algorithm is an iterative process, which terminates when a pre-specified number of generations is produced or when the solution quality is sufficient.

The Genetic Algorithm for the Vehicle Routing Problem described in the article of Baker and Ayechew [1] is implemented and the performance of the algorithm is analysed. The algorithm is applied to four different vehicle routing instances. In this bachelor thesis, the results are analysed and compared to the results described by Baker and Ayechew [1]. We found out that our results are in some cases better than the results mentioned in the article of Baker and Ayechew [1]. However, the time after which the Genetic Algorithm terminates is longer in our case. The main cause for this observation is that we used a different programming language than Baker and Ayechew [1] to implement the Genetic Algorithm.

The algorithm turns out to be adequate for problem instances for which the optimal tours are known. Different variations of the algorithm were tried. The best results were obtained when each of the population members has an equal chance of being selected as a parent. However, not all possible modifications are implemented and there is no evidence that this specific modification performs well for all vehicle routing instances. Besides that, combinations of modifications can be tested. Two or more adaptations together may improve the performance of the algorithm even further.

It turns out that the Hybrid Genetic Algorithm is competitive with tabu search and simulating annealing in terms of solution quality. Tabu search and simulating annealing are methods that have been used to obtain the best known results for benchmark Vehicle Routing Problems. The computing time of the Genetic Algorithm is much longer than those of the other two methods mentioned above. The use of Matlab as the implementing environment can be an explanation for this observation and our advice is to use another programming language, such as C.

Table of contents

1. Introduction

This chapter discusses the practical relevance of the Vehicle Routing Problem and summarizes our current understanding of this type of logistic problem. This chapter is used to relate our study to previous literature. The aim of this research is to verify whether the Genetic Algorithm for the Vehicle Routing Problem is competitive to other, well-known heuristics in terms of solution quality and computing time.

1.1 The Vehicle Routing Problem

The classical Vehicle Routing Problem (VRP) is a combinatorial optimization problem, introduced by Dantzig and Ramser [2], in which customers with known demand must be supplied by a fleet of vehicles. A route is defined as a sequence of visited customers by a certain vehicle, which starts and ends at the depot. The set of routes of all vehicles in the fleet together is denoted as the tour. The demand of a customer is known in advance and must be fully supplied in one visit. A cost is associated for travelling between the depot and the customers. The goal of the basic Vehicle Routing Problem is to minimise the total cost of the tour, while serving all customers.

The classical Vehicle Routing Problem is a generalization of the Travelling Salesman Problem (TSP). The goal of the Travelling Salesman Problem is to find the shortest route for one vehicle that visits each of the customers exactly once. The total cost is proportional to the distance, so the shortest path results in the least possible cost. The Multiple Travelling Salesmen Problem (MTSP) allows for multiple vehicles and is therefore equal to the classical Vehicle Routing Problem.

1.2 Variants

The Capacitated Vehicle Routing Problem is the most basic variant of the standard Vehicle Routing Problem. In this variant, the vehicles have a maximum capacity. A route is feasible when the total demand of the customers supplied by one vehicle does not exceed the maximum capacity. In this study, we only investigate the Homogeneous Capacitated Vehicle Routing Problem, in which each vehicle is assumed to be similar. All vehicles in the fleet have the same capacity Q. That assumption is generally made for Vehicle Routing Problems. Some indirect restrictions comes with this limitation on the carrying capacity. The demand of a customer never exceeds the capacity of a vehicle, because the demand of each customer must be delivered in one visit. The total demand of all customers may not exceed the capacity of all vehicles together. Different vehicle types are included in the fleet in the Heterogeneous Capacitated Vehicle Routing Problem, each with their own capacity. Different vehicle types have different fixed and variable costs, dependent on the specifications of the truck.

The variation of the Homogeneous Capacitated Vehicle Routing Problem that is used in this study is the incorporation of a limited travelling distance for each of the vehicles in the fleet. This type of problem is called the distance constrained version of the Capacitated Vehicle Routing Problem. An extra restriction is added to the constraints that must hold to attain feasibility. In this case, feasibility means that the total demand of all customers assigned to a vehicle is less than or equal to the capacity of that vehicle and that the total travelling distance necessary to supply all customers that are assigned to a specific vehicle does not exceed the maximum travelling distance per vehicle. Due to the fact that the total distance of a route often influences the duration of the route, this is a relevant extension when each carrier may only work a pre-specified number of hours. A drop allowance, which can be seen as the time needed to unload the goods at the customer, is also included. A drop allowance reduces the distance that a vehicle can travel, because it is added to the travel distance of a vehicle. The value of the drop allowance can be equal to zero.

Some delivery companies have multiple depots from which they can supply their customers. The Multi-Depot Vehicle Routing Problem (MDVRP) can be modelled as a set of independent Vehicle Routing Problems when the customers are clustered around depots. Otherwise, the Multiple Depot Vehicle Routing Problem should be solved, see [3]. Most algorithms that are developed for this class of problems are based on meta-heuristics. Meta-heuristics applied to Vehicle Routing Problems are top-level general strategies that guides other heuristics to search for feasible tours. Two well-known meta-heuristics that are used to solve the Vehicle Routing Problem and its variants are tabu search and simulated annealing.

The Vehicle Routing Problem with Pick-up and Delivery (VRPPD) is a routing problem in which customers can return commodities to the vehicle which must be brought to other customers or to the depot. The goods that customers return to the vehicle must fit into it. Priority is given to the pick-up activity when the commodities must be transported to known delivery locations. Several variants have been proposed. The Vehicle Routing Problem with Backhauls (VRPB) is a pick-up and delivery problem with the additional restriction that all goods must be delivered before commodities can be picked up by customers, because the vehicle is filled 'Last-In, First-Out', see Dethloff [4]. One well-known application of a pick-up and delivery problem is the delivery of goods in reusable boxes. Customers pay a deposit per box, which will be repaid when the customer returns the empty box to the delivery company.

The classical Capacitated Vehicle Routing Problem does not take into account the concept of time windows. In reality, each customer has stated a number of time periods in which the demand of that customer must be fulfilled. The introduction of time windows makes the problem far more complex, as the flexibility of delivery companies is decreased. More details about the Capacitated Vehicle Routing Problem with Time Windows (CVRPTW) are provided by Beasley and Christofides [5].

1.3 Solving methods

The Capacitated Vehicle Routing Problem covers an important field in transportation and logistics, as many distribution companies have to deal with this type of problem every day. Transportation costs are a significant part of the total logistics cost. Due to increases in the price of fuel, this proportion continues growing over the years. There are a number of transportation strategy that can be used to reduce costs, e.g. by reducing the number of carriers, the consolidation of shipments or efficient routing. Reducing the transportation cost is an important activity, both at the micro level and at the macro level.

It is important that a good tour can be found within a reasonable time limit. Unfortunately, there are nowadays no exact algorithms that can solve every vehicle routing instance. Some efficient algorithms work well up to a certain amount of customers and vehicles. Some instances with a large number of customers and vehicles are easy and can be solved exactly. Two exact formulations will be discussed in Section 2.2. More details can be found in the article of Toth and Vigo [6]. Heuristics are used to solve vehicle routing instances which cannot be solved with an efficient exact algorithm. A lot of research is done to improve the algorithms that are used to solve such routing instances.

The Capacitated Vehicle Routing Problem is an NP-hard problem, which means that the fastest algorithms take exponential time, when P = NP. Heuristics are used to find a near-optimal, feasible solution within a reasonable time limit. The tabu search implementations of Taillard [7] and Rochat and Taillard [8] have obtained the best known results to benchmark Vehicle Routing Problems. Similar results are obtained with simulated annealing [9]. The objective of this bachelor thesis is to implement the Genetic Algorithm for the Capacitated Vehicle Routing Problem described by Baker and Ayechew [1] and to verify the results that are described in that paper. This routing heuristic is introduced in Subsection 1.3.1. Variations of the algorithm are tried that may improve the performance of the search heuristic.

1.3.1 The Genetic Algorithm

The general idea of a Genetic Algorithm is related to the theory of evolution by means of natural selection, stated by the English naturalist and geologist Charles Darwin (1809–1882). An initial population is generated and this population is improved, such that the fitness value of the individuals in the population becomes better through different generations. Individuals, also called candidate solutions, that are better adapted to their environment have a higher chance to survive. New individuals are generated by crossover and simple mutations. The weaker individuals are removed from the population, which will improve the overall fitness of the population. The number of individuals in the population is constant over the different generations and duplicating members are not allowed. There must be a balance between the genetic quality and the diversity of the individuals

in the population to support efficient search. The Genetic Algorithm is an iterative process, which terminates when a pre-specified number of generations is produced or when the solution quality is sufficient.

Baker and Ayechew [1] made use of fourteen different vehicle routing instances to analyse the quality of their Genetic Algorithm. Our objective is to reproduce all steps that are carried out in the article and to verify the results. We are interested in the number of vehicles needed to supply all customers and the associated minimum total cost. As mentioned in the article of Baker and Ayechew [1], the objective of the algorithm is often taken to be equivalent to minimising the total movement of vehicles, or to minimising the number of vehicles in the fleet and then minimising the total distance for this number of vehicles.

The purchase of a vehicle and the associated depreciation are called fixed costs. The salary cost for drivers is also a fixed cost. The variable costs are, among other things, fuel consumption and maintenance. The fixed costs are relatively high in comparison with the variable costs per kilometre. As one can imagine, the number of vehicles necessary to fulfil all demand must be minimised. In that way, the fixed costs are minimised. Given the number of vehicles necessary to fulfil all demand, the total travel distance must be minimised, such that all customers are supplied and the variable costs are minimised.

The rest of this report is organized as follows: In Chapter 2, the problem definition of the distance constrained version of the Capacitated Vehicle Routing Problem is provided, together with two different flow formulations. These flow formulations can be used to find the exact solution of asymmetric and symmetric vehicle routing instances. As mentioned above, most of the instances cannot be solved with an exact algorithm. Important variables that are used throughout this paper are introduced in Chapter 2. Chapter 3 is used to explain the Genetic Algorithm implemented by Baker and Ayechew [1] in more detail. Section 3.1 describes the Pure Genetic Algorithm, followed by a description of the neighbourhood search methods of the Hybrid Genetic Algorithm in Section 3.2. The modifications of the algorithm that were tested are explained in Section 3.3. These modifications may improve the performance of the routing heuristic. In Chapter 4, the data which are used to evaluate the Genetic Algorithm are described. The structure of these data and important features are highlighted. The results obtained with the Genetic Algorithm are analysed in Chapter 5. This chapter also evaluates the performance of the algorithm for instances for which the optimal solutions are known. Section 5.3 analyses the results obtained with each of the modifications. After this chapter, the conclusion and recommendations for future research are given. This paper ends with the bibliography, the list of figures and tables and the appendix.

2. The distance constrained version of the Capacitated Vehicle Routing Problem

This chapter is used to give a formal definition of the distance constrained version of the Capacitated Vehicle Routing Problem and to introduce important variables that are used throughout this study. Two flow formulations are provided which can be used to find an exact solution for the distance constrained version of the Capacitated Vehicle Routing Problem.

2.1 Problem definition

Let V = {*0, 1, ..., n*} be the set with all customer locations, where *0* represents the depot. There are *n* customers with deterministic demand that must be satisfied. The fleet size, or the number of vehicles, is equal to *m*. Between every pair of customers and between each customer and the depot, there is a minimum distance route. One option is to assume that the shortest travel distance from customer location *i* to customer location *j* is the same as the shortest travel distance from *j* to *i*, for all *i*, *j* ∈ V. Let E = {(*i,j*)| *i,j* ∈ V; *i* < j} be the edge set between each pair of vertices, also called the road set. The complete graph is denoted as G = (V,E). The shortest distance corresponding to edge (*i,j*) is the nonnegative cost c_{ij} and $c_{ij} = c_{ji}$ for all *i,j* ∈ V. The distance matrix generally satisfies the triangle inequality: $c_{ij} \leq c_{iw} + c_{wj}$ for $0 \leq i, j, w \leq n$. Each customer *i*, where *i* ∈ V \ {*0*}, has a known demand quantity q_i. Each vehicle has the same carrying capacity, denoted by Q. The total demand of all customers supplied by vehicle *k*, for *k* ∈ {1, ..., *m*}, may not exceed the capacity of that vehicle. This type of problem is called Symmetric Capacitated Vehicle Routing Problem (SCVRP).

The classical Capacitated Vehicle Routing Problem is extended with a restriction on the travelling distance per vehicle. Each vehicle in the fleet has a maximum travelling distance D. A drop allowance is associated with each customer and it is denoted by ρ_i for *i* ∈ V \ {*0*}. A drop allowance can be seen as the time that is needed to unload the goods at a customer and this value is added to the distance travelled by the vehicle. The drop allowance can dependent on the amount of demand of the customer, for example. Drop allowances reduce the distance that a vehicle can travel when they are larger than zero. In that case, they can reduce the number of possible combinations of customers that can be supplied by one of the vehicles in the fleet, as represented in Figure 2.1.

Without any drop allowance, customer four in Figure 2.1 could also has been supplied by vehicle one. The total demand of the customers is ninety and the travelling distance is also ninety in this specific case. However, when drop allowances need to be incorporated, the travelling distance is 130 when all customers are supplied by the same vehicle. The maximum distance that this vehicle can travel is one hundred, which leads to the conclusion that it is not possible to supply all customers when a drop allowance of ten is associated with each customer.

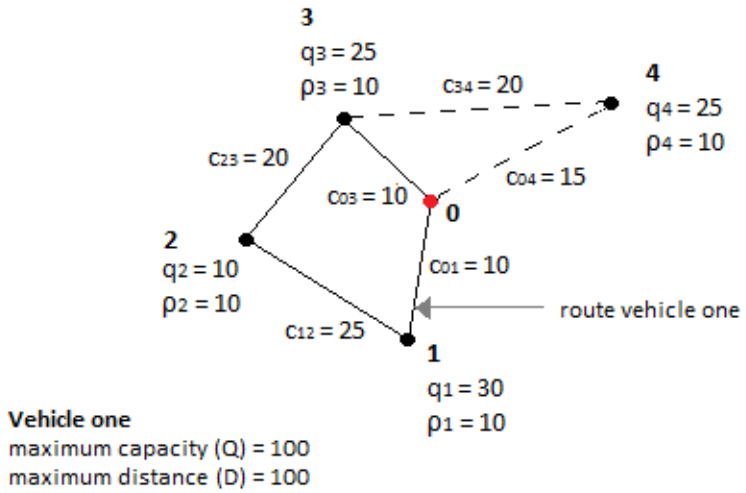


Figure 2.1: Influence of the drop allowance on the travelling distance.

The total distance travelled to supply all customers assigned to vehicle *k*, together with the sum of the drop allowances associated with these customers, must be smaller or equal to the maximum travel distance D. Our goal is to minimise the total cost. Figure 2.2 shows the complete graph G = (V,E) of a symmetric capacitated vehicle routing instance with five customers.

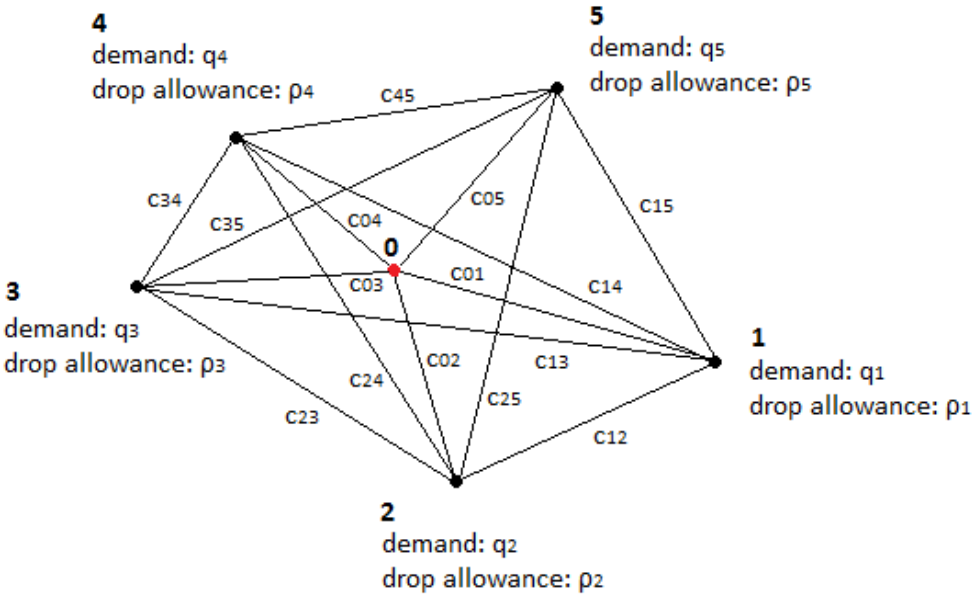


Figure 2.2: Complete graph of a small, symmetric capacitated vehicle routing instance, where the customers are randomly distributed around the depot. A drop allowance is associated with each customer.

2.2 Vehicle flow formulations

This section describes two vehicle flow formulations of the Capacitated Vehicle Routing Problem provided by Toth and Vigo [10]. The first formulation can be used to find the optimal tour for both symmetric and asymmetric vehicle routing instances. The second formulation can only be used for symmetric instances and reduces the number of decisions variables.

The three-index formulation is used to explicitly indicate the vehicle that traverses an arc. The decision variable x_{ijk} is equal to one if customer *j* is the successor of customer *i* on route *k,* for *i,j* ∈ V and *k* ∈ {1, ..., *m*} and zero otherwise. The decision variable y_{ik} equals one if customer *i* is assigned to vehicle *k*, for *i* ∈ V and *k* ∈ {1, ..., *m*} and zero otherwise. With this formulation, we are able to impose more involved constraints on the routes than in a two-index formulation. In the last case, x_{ij} is one if customer *j* is the successor of customer *i*, but we do not have information about the vehicle that supplies these two customers. The three-index formulation for the asymmetric and the symmetric Capacitated Vehicle Routing Problem (ACVRP/SCVRP) is given in the following. This formulation assumes that the minimum number of vehicles necessary to supply all customers without any constraint violation is known.

$$(ACVRP/SCVRP) \quad \min \sum_{i \in V} \sum_{j \in V} c_{ij} \sum_{k=1}^{m} x_{ijk} \tag{1.1}$$

subject to

$$\sum_{k=1}^{m} y_{ik} = 1 \qquad for\ all\ i \in V \backslash \{0\} \tag{1.2}$$

$$\sum_{k=1}^{m} y_{0k} = m \tag{1.3}$$

$$\sum_{j \in V} x_{ijk} = \sum_{j \in V} x_{jik} = y_{ik} \qquad for\ all\ i \in V, k = 1, \dots, m \tag{1.4}$$

$$\sum_{i \in V} q_i y_{ik} \leq Q \qquad for\ all\ k = 1, \dots, m \tag{1.5}$$

$$\sum_{i \in V} \sum_{j \in V} c_{ij} x_{ijk} + \sum_{i \in V} \rho_i y_{ik} \leq D \quad for\ all\ k = 1, \dots, m \tag{1.6}$$

$$u_{ik} - u_{jk} + Q x_{ijk} \leq Q - q_j \qquad for\ all\ i, j \in V \backslash \{0\}, i \neq j, \tag{1.7}$$

$$such\ that\ q_i + q_j \leq Q, k = 1, \dots, m$$

$$q_i \leq u_{ik} \leq Q \qquad for\ all\ i \in V\backslash\{0\}, k = 1, \dots, m \qquad (1.8)$$

$$y_{ik} \in \{0,1\} \qquad for\ all\ i \in V, k = 1, \dots, m \qquad (1.9)$$

$$x_{ijk} \in \{0,1\} \qquad for\ all\ i,j \in V, k = 1, \dots, m \qquad (1.10)$$

The objective is to minimise the total distance travelled by all vehicles together. Constraints (1.2) - (1.3) impose that each customer is visited exactly ones and that *m* vehicles leave the depot, respectively. Constraints (1.4) impose that each customer on route $k \in \{1, ..., m\}$ must have one predecessor and one successor, which can be the depot or another customer. Constraints (1.5) impose that the total demand supplied by one vehicle is less than or equal to its capacity, whereas constraints (1.6) are the distance restrictions for each vehicle $k \in \{1, ..., m\}$. The last constraints take into account both the vehicle movements and the drop allowance associated with the customers. To obtain a flow formulation for the classical Capacitated Vehicle Routing Problem, these last constraints should be removed. Constraints (1.7) - (1.8) eliminate possible sub tours. Both decision variables y_{ik} and x_{ijk} are binary variables.

In the symmetric case, we do not care about whether customer *j* is the successor of customer *i* or the predecessor of customer *i*. In that case, the three-index variable x_{ijk} can be replaced by the decision variable x_{ek}, associated with the undirected edges $e \in E$ and $k \in \{1, ..., m\}$. Let λ(i) denote the set of edges $e \in E$ that have one endpoints at customer $i \in V$. Special conditions holds for x_{ek} when $e \in \lambda(0)$. If $e \notin \lambda(0)$, then $x_{ek} \in \{0,1\}$, whereas if $e \in \lambda(0)$, then $x_{ek} \in \{0, 1, 2\}$. This allows for the possibility that only one customer is served by a specific vehicle. Only in that case, a specific edge is used twice. The distance matrix is symmetric, in the sense that $c_{ij} = c_{ji}$ for all $i,j \in V$. Let c_e be the travel distance of edge $e \in E$. The symmetric version of the three-index formulation is represented by the following. Again, this formulation assumes that the minimum number of vehicles necessary to supply all customers without any constraint violation is known.

$$(SCVRP) \quad \min \sum_{e \in E} c_e \sum_{k=1}^{m} x_{ek} \qquad (1.11)$$

subject to

$$\sum_{k=1}^{m} y_{ik} = 1 \qquad for\ all\ i \in V\backslash\{0\} \qquad (1.12)$$

$$\sum_{k=1}^{m} y_{0k} = m \qquad (1.13)$$

$$\sum_{e \in \lambda(i)} x_{ek} = 2y_{ik} \qquad for\ all\ i \in V, k = 1, \dots, m \qquad (1.14)$$

$$\sum_{i \in V} q_i y_{ik} \leq Q \qquad for\ all\ k = 1, \dots, m \qquad (1.15)$$

$$\sum_{e \in E} c_e x_{ek} + \sum_{i \in V} \rho_i y_{ik} \leq D \qquad for\ all\ k = 1, \dots, m \qquad (1.16)$$

$$\sum_{e \in \lambda(S)} x_{ek} \geq 2y_{hk} \qquad for\ all\ S \subseteq V\backslash\{0\}, h \in S, k = 1, \dots, m \qquad (1.17)$$

$$x_{ek} \leq |S| - 1 \qquad for\ all\ S \subseteq V\backslash\{0\},\ |S| \geq 2, k = 1, \dots, m \qquad (1.18)$$

$$y_{ik} \in \{0, 1\} \qquad for\ all\ i \in V, k = 1, \dots, m \qquad (1.19)$$

$$x_{ek} \in \{0, 1\} \qquad for\ all\ e \notin \lambda(0), k = 1, \dots, m \qquad (1.20)$$

$$x_{ek} \in \{0, 1, 2\} \qquad for\ all\ e \in \lambda(0), k = 1, \dots, m \qquad (1.21)$$

The objective is to minimise the total distance travelled by all vehicles together. Constraints (1.12) - (1.13) impose that each customer is visited exactly ones and that *m* vehicles leave the depot, respectively. Constraints (1.14) impose that each vehicle that enters a given customer must also leave that customer. Constraints (1.15) impose that the total demand supplied by one vehicle is less than or equal to its capacity, whereas constraints (1.16) are the distance restriction for each vehicle k. To obtain a flow formulation for the classical Capacitated Vehicle Routing Problem, these last constraints should be removed. Constraints (1.17) impose the connectivity of the route performed by vehicle k, whereas constraints (1.18) eliminate possible sub tours. All variables y_{ik} are binary variables. Constraints (1.20) - (1.21) are explained before. More vehicle flow formulations can be found in the book edited by Toth and Vigo [10].

The two mixed-integer program formulations represented above can both be used to solve symmetric capacitated vehicle routing instances with distance restrictions. Also, routing instances without distance restrictions and capacity limits per vehicle can be solved with the formulations mentioned above. Only the first formulation is adequate to solve asymmetric vehicle routing instances. The second formulation reduces the number of decision variables needed compared to the first formulation, which will speed up the process. Symmetric instance can best be solved with the second formulation. Both formulations will give the same optimal value for symmetric instances, but are not efficient enough to solve every possible routing instance. For those instances that cannot be solved exactly, heuristics are developed to find near-optimal, feasible solutions within a reasonable time limit.

3. Methodology

This chapter is used to describe the Genetic Algorithm for the Vehicle Routing Problem in more detail. The Pure Genetic Algorithm is described in Section 3.1, including technical information about the implementation of this algorithm for the Capacitated Vehicle Routing Problem with distance limits. Section 3.2 provides a description of the Genetic Algorithm Hybridised with neighbourhood search. The last section of this chapter discusses the modifications of the Genetic Algorithm that are tried.

3.1 The Pure Genetic Algorithm

The overall goal of the Genetic Algorithm is to emulate the natural selection process. The population of candidate solutions will evolve to a population with better solutions, by repeatedly applying mutations and crossovers. Natural selection implies that individuals who are better adjusted to their environment have a higher change to survive than weaker individuals. The same mechanism can be used to find a solution for the Vehicle Routing Problem and its variations. Figure 3.1 shows a simplified flow chart of the most general Genetic Algorithm.

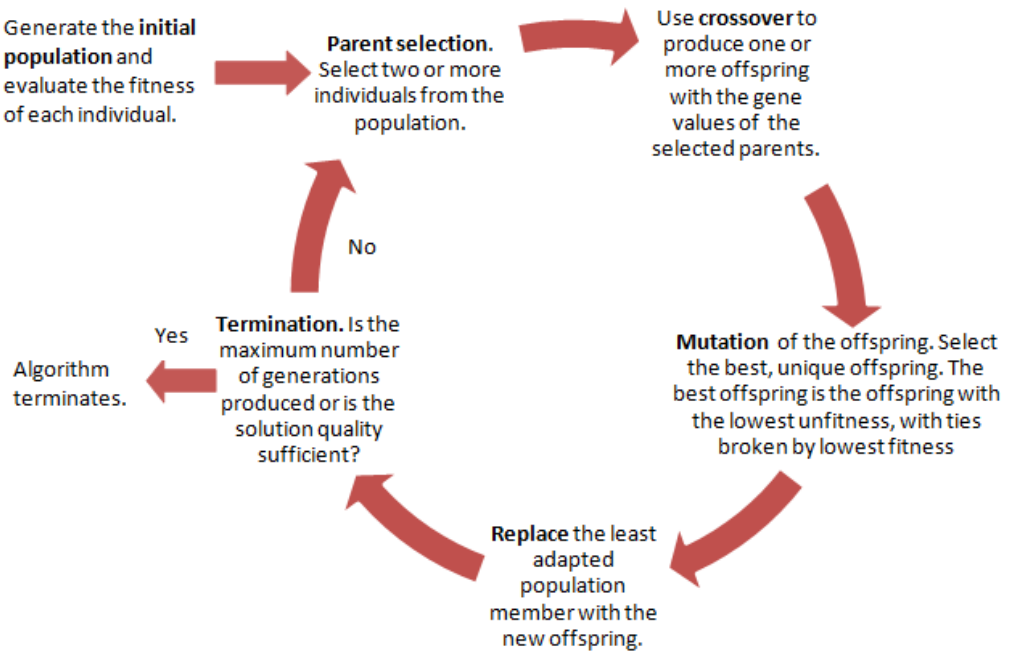


Figure 3.1: Simplified flow chart of a general Genetic Algorithm.

The first step of the Pure Genetic Algorithm is to generate an initial population. This can be done in many different ways, for example randomly or with a specific approach. Each individual is represented by a chromosome of length n, the number of different customers. The value of the i-th gene of this chromosome denotes the vehicle number by which the i-th customer is supplied. This number is in the range [1, m], where m is the total number of vehicles in the fleet. The explicit order

in which the customers are served is found by the solving the Travelling Salesman Problem for each of the vehicles. This ordering cannot be abstracted directly from the chromosomes. The population members are used to create new offspring, by applying crossover to selected parent solutions. With some chance, mutations are applied to the newly created children. A replacement method updates the current population. Baker and Ayechew [1] made use of the Steady State Genetic Algorithm, which means that the new population consists of the best individuals rather than only the generated children (Generational Genetic Algorithm). The population size is fixed and the algorithm terminates when a pre-specified number of generations is produced or when the solution quality is sufficient.

The total distance travelled by all vehicles is denoted as the fitness of the population member and the sum of any excess distance and capacity on a vehicle, expressed as a percentage of the maximum travelling distance and capacity respectively, is denoted as the unfitness of the individual. An individual with a lower unfitness value is better than an individual with a higher unfitness value, irrespective of the fitness values of both individuals. The best individual in a population is the member with the lowest unfitness value, with ties broken by lowest fitness value. Only individuals with an unfitness of zero are feasible, in the sense that no constraint violations occur in that case.

3.1.1 Creation of the initial population

The first step of the Genetic Algorithm is to create an initial population. As described by Baker and Ayechew [1], different methods can be used to generate the members of the initial population. One possible option is to generate half of the initial population with the sweep approach of Gillet and Miller [11] and the other half of the initial population with the generalised assignment approach of Fisher and Jaikumar [12].

3.1.1.1 The sweep approach

The first method used to generate individuals for the initial population is based on the sweep approach of Gillett and Miller [11]. The general idea is to sort customers in accordance with a sorting method, which is different for instances where customers are randomly distributed around the depot than for routing instances where customers are grouped in clusters. Customers are sorted according to increasing order of polar angle when their locations are randomly distributed around the depot. Figure 3.2 shows the polar angle θ of the red point. It is simply the angle between the point and the x-axis. The nearest neighbour solution to the Travelling Salesman Problem is used to sort the customers when they are located in clusters. After the sorting phase, a customer is chosen at random to start the sweep process. The demand of the next ordered customer is supplied by the same vehicle as long as there are no constraint violations.

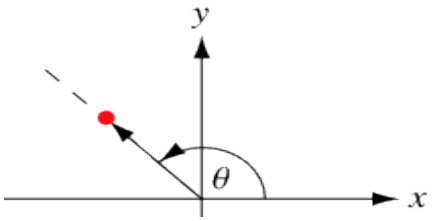


Figure 3.2: The polar angle θ, see Weisstein [13].

After allocating one or more customers to a vehicle, adding the next customer will cause a constraint violation. Table 3.1 shows the allocating rules that we have used to determine whether this customer should be assigned to the vehicle or not.

The problem tightness refers to the percentage of the total vehicle capacity that must be used to satisfy all demand of customers, e.g. when there are six vehicles with a capacity of one hundred and the total demand of all customers is equal to five hundred, the problem tightness is about 0.83 (= 500 / 600). The value of R_C is the remaining capacity of the vehicle before adding the customer that causes a violation, expressed as a proportion of the demand of this customer. Suppose that each vehicle has a capacity of one hundred and that the total demand of all customers already assigned to a vehicle is ninety. When the next ordered customer has a demand of twenty, the value of R_c is equal to 0.5 (= (100 – 90) / 20). R_d is the additional distance that the vehicle could have travelled before adding the last customer, expressed as a proportion of the additional distance required to include the last customer. Suppose that the maximum travel distance per vehicle is one hundred and that the total travel distance so far is ninety-five. When the additional distance required to add the next ordered customer is seven, the value of R_d is round off 0.71 (= 5 / 7).

When the problem tightness is small, it is not a big deal that some vehicle have spare capacity. In that case, the last customer is only included in the sweep when the value for R_c or R_d is close to one, such that there is only a small constraint violation.

Table 3.1: Conditions for including the last customer in sweep during the sweep approach.

	Capacity constraints only	**Capacity and distance constraints**
Not tight	tightness < 0.95, $R_c \geq 0.9$	tightness < 0.8, $R_d \geq 0.9$
Tight	tightness ≥ 0.95, $R_c \geq 0.75$	tightness ≥ 0.8, $R_d \geq 0.75$

The sweep approach continues by selecting the next ordered customer who is not supplied by a vehicle, allocating it to an empty vehicle when there is one available and repeating the same procedure. When at least one customer cannot be assigned to a vehicle, this individual is not accepted and the process is started over again. After every fifty unsuccessful tries to generate an unique individual, the values for R_c and R_d are reduced with 0.01. This means that the last customer in sweep is included with a higher probability, because a larger constraint violation is allowed. Lower

values of R_c and R_d make it more likely that individuals with an unfitness greater than zero are created. The main ideas of the sweep approach are summarised in pseudo code in Algorithm 3.1.

Algorithm 3.1: ***Sweep approach for the generation of the initial population***

1. Initialization

   s_1 — Number of individual that must be generated, depends on the number of customers that must be supplied.

   $R_{c,k,r} = 0$ — Remaining capacity of vehicle *k* for individual *r*, before adding the last customer, expressed as a proportion of the demand of this customer. Here, *k* = 1, ..., *m* and *r* = 1, ..., s_1.

   $R_{d,k,r} = 0$ — Remaining distance that vehicle *k* of individual *r* could have travelled before adding the last customer, expressed as a proportion of the additional distance required to include that customer. Here, *k* = 1, ..., *m* and *r* = 1, ..., s_1.

   v — Counter variable that saves the number of successive unsuccessful creations.

   $P = \frac{\sum_{i=1}^{n} q_i}{mQ}$ — Problem tightness. This value is the same for all population individuals.

2. Sorting step

   Sort the customers according to increasing order of polar angle or the nearest neighbour solution to the Travelling Salesman Problem, dependent on the customer locations.

3. Generating step

   Select one customer at random, who is allocated to the first vehicle.

   repeat

   repeat

   Add the next ordered customer to the current vehicle.

   until a constraint violation occurs

   Calculated R_c and, when the problem has also distance constraints, R_d.

   if problem has only capacity constraints **then**

   if $P < 0.95$ **then** Include last customer in sweep if $R_c \geq 0.9$

   else Include last customer in sweep if $R_c \geq 0.75$

   end if

   else

   if $P < 0.8$ **then** Include last customer in sweep if $R_d \geq 0.9$

   else Include last customer in sweep if $R_d \geq 0.75$

   end if

   end if

   When there are still customers not assigned to a vehicle, assign the customer where the last one left off to a new vehicle.

   until all customers are assigned to a vehicle or all vehicles are used

   Check whether this creation is successful, otherwise v ← v + 1.

   If v = 50 **then**

   $R_d = R_d - 0.01$ and $R_c = R_c - 0.01$

   v ← 0

   end

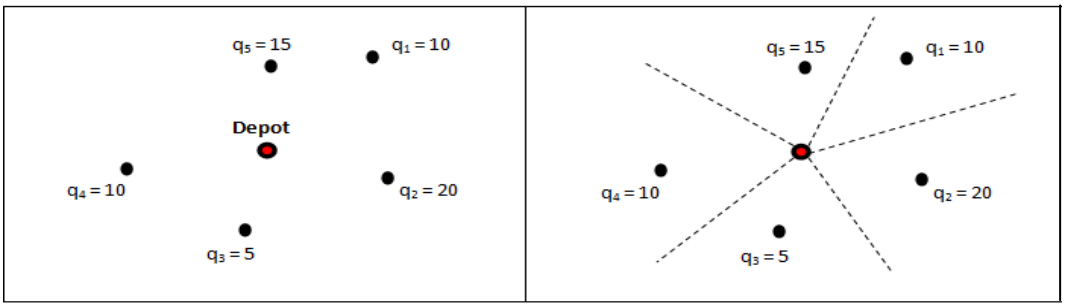
   Repeat this step as long as the newly created individual duplicates another population member.

4. Repeat step 3 s_1 times. Save the created individual after each iteration.

3.1.1.2 The generalised assignment approach

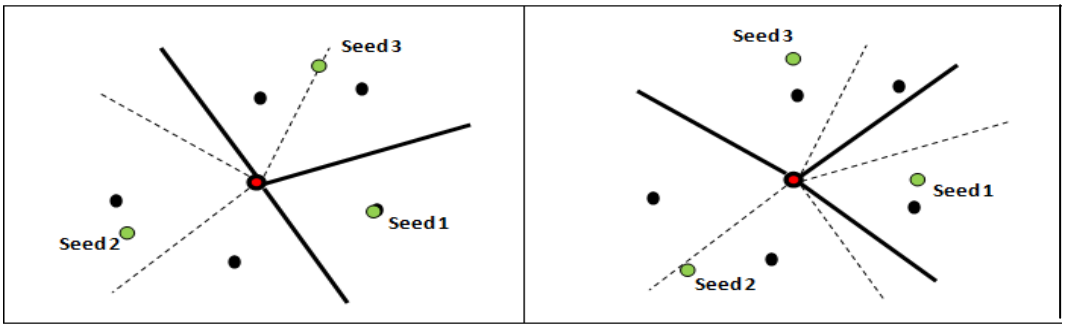
The second method to generate individuals for the initial population makes use of the generalised assignment approach of Fisher and Jaikumar [12]. Assuming that all customers are distributed around the depot, the first step is to generate so called customer cones. To do so, lines are drawn from the depot that split the angles between neighbouring customers. Vehicle cones are created by combining customer cones and fractions of customer cones, such that there is an equal demand distribution between vehicle cones. A seed is located along each ray that bisects the angle of the corresponding vehicle cone. The distance of each seed from the depot is equal to the maximum of the customer distance for that vehicle cone. Infinitely many different combinations of vehicle cones can be created, such that there is diversity between the generated individuals. An example of the creation of vehicle cones is represented in Figure 3.3.

For each customer, the cost of allocating that customer to seed t, where $t \in \{n + 1, ..., n + m\}$, is calculated. For customer i, this cost is approximated by $f_{ik} = d_{oi} + d_{ik} - d_{0k}$. Here, d_{hl} is a symmetric distance matrix. The value of element (h,l) is the distance between sweep or customer h and sweep or customer l, where $h,l \in V \cup \{n + 1, ..., n + m\}$. Say f_{ia} and f_{ib} are the smallest and second smallest seed allocation costs for customer i, where $a,b \in \{1, ..., m\}$ and $a \neq b$. Customer i is allocated to its best seed with a probability of $f_{ib} / (f_{ia} + f_{ib})$ and to its second best seed with a probability of $f_{ia} / (f_{ia} + f_{ib})$.



Problem, each customer has a known demand which must be satisfied.

Create customer cones by drawing rays fromt he depot that bisect the angles between adjacent customers according to the order in which they are sorted.



Two examples of how the vehicle cones can be formed. Each vehicle cone has an equal demand of twenty goods. Of course, multiple other combinations of vehicle cones are possible, such that there is diversity between the generated individuals.

Figure 3.3: Process to generate customer cones and vehicle cones.

After allocating all customers to the seeds, a travelling salesman heuristic is used to make the vehicle routes. Individual vehicle routes are found using the 2-optimal method, followed by a 3-optimal refinement. Both techniques are special cases of the k-optimal method. The *k*-optimal method removes *k* mutually disjoint edges of a given tour and reassembles the remaining fragments into a tour. It must be taken into account that sub tours are not allowed. The main ideas of the generalised assignment approach are summarised in pseudo code in Algorithm 3.2.

Algorithm 3.2: ***Generalised assignment approach for generation of the initial population***

1. Initialization

   s_2 Number of individual that must be generated, depends on the number of customers that must be supplied.

2. Cone step

   Create customer cones by drawing lines from the depot that split the angles between neighbour customers. The demand per vehicle cone is equal to $\sum_{i=1}^{n} q_i \ /m$. So, make *m* vehicle cones with a demand of $\sum_{i=1}^{n} q_i \ /m$ each, by aggregating (fractions of) customer cones.

3. Seed step

   Locate in each vehicle cone a seed, which lies along the ray that bisects the angle of the vehicle cone. The distance between the seed and the depot is equal to the maximum of the customer distances for that vehicle cone. Number the seeds from *n* + 1 to *n* + *m*.

4. Initialization

   d_{hl} Symmetric distance matrix, with 0 representing the depot. The value of element (*h,l*) is the distance between sweep or customer *h* and sweep or customer *l*, where *h,l* ∈ V ∪ {*n* + 1, ..., *n* + *m*}.

5. Allocating step

   for *i* = 1, …, *n*

   Calculate the cost of allocating customer *i* to seed *k*, $f_{ik} = d_{oi} + d_{ik} - d_{0k}$, for *k* = 1, …, *m*. Determine the smallest and second smallest allocation costs, say f_{ia} and f_{ib} respectively. Generate a random variable U, which is uniformly distributed over the interval [0,1].

   if $U \leq f_{ib} / (f_{ia} + f_{ib})$ **then**

   Allocate customer *i* to its best seed.

   else

   Allocate customer *i* to its second best seed.

   end if

   end for

   Repeat this step as long as the newly created individual duplicates another population member.

6. Routing step

   Route the customers allocated to each seed by using a travelling salesman heuristic. Due to the fact that this step is performed quite often, a heuristic is used to find a good approximation of the optimal routes.

7. Repeat the process, starting from step 2, s_2 times. Save the created individual after each iteration.

3.1.2 Creation of new offspring

After the generation of the initial population, new individuals can be produced given the population members. First of all, we renumber the vehicles in such a way that each vehicle supplies customers in approximately the same region for each of the candidate solutions. This can be done, for each population member, by computing the average of both coordinates for customers supplied by each vehicle. For each member, this gives the points $(\overline{x_k}, \overline{y_k})$, where k = 1, ..., m. The polar angles of these points are used to number the vehicles, with vehicle one related to the angle closest to zero. The other vehicles are numbered in order of increasing polar angle. When one route lies entirely within another route, the inner route gets the lowest vehicle number of the two. This can be done as follows. Given that we have already calculated the average coordinate points of all customers visited by each vehicle, it is easy to calculated the distance from the depot to $(\overline{x_{k+1}}, \overline{y_{k+1}})$ and the distance from the depot to $(\overline{x_k}, \overline{y_k})$. When the first distance is less than half of the second distance, and the difference between the polar angles is less than 180°/m, the vehicle numbers are accorded such that the inner route has a lower number than the outer route.

A binary tournament method, which is discussed in this paragraph, is used to select two parent solutions from the population. To select a parent, two individuals are randomly chosen from the population and the one with the lowest fitness is selected. The parents are used to produce new offspring. Before creating new offspring, the vehicles of parent one must be re-numbered to make the angles for the first vehicles match more closely between the two parents. Two children are generated, by randomly selecting the part of the chromosomes of the selected individuals that should be interchanged. This process is illustrated in Figure 3.4.

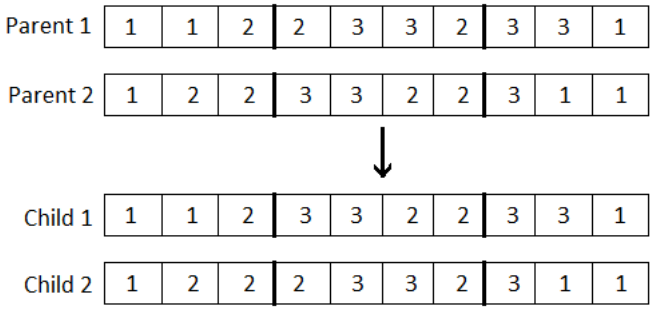


Figure 3.4: Crossover process to create new offspring.

As described by Baker and Ayechew [1], the performance of the Genetic Algorithm is improved when a small mutation is applied to the new offspring. Two customers are selected at random and their gene values are interchanged when they are different. This process is illustrated in Figure 3.5, for one offspring created in Figure 3.4.

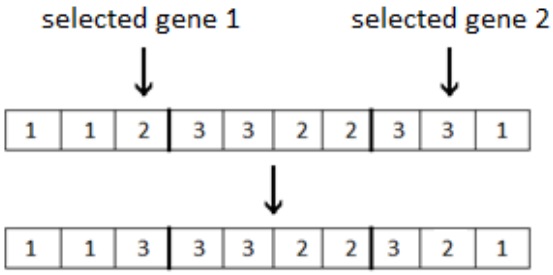


Figure 3.5: Simple mutation. Two customers are selected at random and their gene values are interchanged.

After applying this simple mutation to both offspring, we must check whether or not the newly created offspring duplicate existing population members. When both children duplicate another population member, this generation is unsuccessful and new offspring must be generated until an offspring is created that does not duplicate an existing member. A successful generation refers to a generation in which at least one of the children does not duplicate one of the population members. The best unique offspring is selected and the vehicles are renumbered with the procedure described in this subsection. Algorithm 3.3 provides the pseudo code for this reproduction phase.

Algorithm 3.3: ***Reproductive process to create a new offspring***

1. Initialization

   $P = \{P_1, P_2, \ldots, P_{s1+s2}\}$ Set of all population members. Note that there are $s_1 + s_2$ individuals, because s_1 individuals are generated with the sweep approach (algorithm 3.1) and s_1 individuals are generated with the generalised assignment approach (algorithm 3.2).

   $t = 2$ Number of individuals that must be chosen from the population during the tournament method. Can easily be adjusted to increase or decrease the tournament size.

2. Fitness step, only after the generation of the initial population

   for $p = 1, \ldots, s_1 + s_2$

   Calculate the fitness value of population individual p, $f(P_p)$ which is defined as the travel distance of this individual. Also calculate the unfitness of this candidate solution, $u(P_p)$.

   end for

3. Tournament method

   First select parent one, set $z \leftarrow 1$.

   repeat

   Choose t individuals at random from the population. The one with the lowest fitness value is chosen as parent z, with ties broken by lowest unfitness.

   $z \leftarrow z + 1$

   until two parents are selected

4. 2-point crossover

   Generate a random integer X_1, which is uniformly distributed over the interval [0,n].

   Generate a random integer X_2, which is uniformly distributed over the interval [0,n].

   Set $U = \max(X_1, X_2)$ and $L = \min(X_1, X_2)$.

   The first offspring contains the gene values from parent one which are to the left of L and to the right of U, together with the gene values from parent two between L and U. The same procedure is used to generate the second offspring, by swapping round the parents.

5. Offspring step

   Modify both offspring by applying a simple mutation to them, in which two genes are selected at random and their values are exchanged. Check whether both newly created individuals duplicate another population member. If so, start again at step 3. Otherwise, select the unique offspring with the best route structure. Repeat the process until an unique offspring is created.

3.1.3 Replacement phase

Due to the fact that the population size is fixed, one of the population members should be removed. The Genetic Algorithm mimics the process of natural selection. This means that the weakest individual should be removed. The population members are subdivided into four sets with respect to the fitness and unfitness of the offspring. This process is called the ranking replacement method, see Beasley and Chu [14].

The total travel distance of an individual can be found by solving the Travelling Salesman Problem for each of the vehicles. The Travelling Salesman Problem makes the actual routes, after which the fitness value for the individual can be calculated. A heuristic is used to make the routes explicit, because solving the Travelling Salesman Problem exactly takes much computation time and this problem must be solved very often during the initialisation and reproduction phase of the algorithm. Individual vehicle routes are found using the 2-optimal method, followed by a 3-optimal refinement. The 2-optimal technique is a special case of the k-optimal method. The *k*-optimal method removes *k* mutually disjoint edges of a given tour and reassembles the remaining fragments into a tour. It must be taken into account that sub tours are not allowed. The 3-optimal heuristic removes three mutually disjoint edges and reconnect the tour with three edges without creating a sub tour. All possible reconnections are tried and the one which yields the lowest cost or travelling distance is accepted.

Four disjoint sets of population members can be created, according to the fitness and unfitness value of the offspring. The population member with the worst unfitness of all members with a fitness value and unfitness value larger or equal than those of the best, unique offspring is removed from the population. However, when there are no such individuals, the population member with the worst unfitness of all members with a fitness value smaller and an unfitness value larger or equal than those of the newly added offspring is removed. When that subset is also empty, check whether there is at least one population member with a fitness value larger or equal and an unfitness value smaller than those of the newly added offspring. The individual with the worst unfitness is replaced. When for all population members both the fitness and unfitness values are smaller than those of the best offspring, the offspring itself is not added to the population. This generation is also called successful, because the best offspring does not duplicate another population member. It simply does not improve the overall quality of the population.

The process of creating new offspring, started from the point of selecting two parents, is repeated until the termination criterion is satisfied. Algorithm 3.4 provides the pseudo code for the replacement phase.

Algorithm 3.4: ***Replacement scheme to remove one individual***

1. Initialization

   $S_1 = \{P_p \in P: f(P_p) \geq f(c), u(P_p) \geq u(c)\} = \emptyset$

   $S_2 = \{P_p \in P: f(P_p) < f(c), u(P_p) \geq u(c)\} = \emptyset$

   $S_3 = \{P_p \in P: f(P_p) \geq f(c), u(P_p) < u(c)\} = \emptyset$

   $S_4 = \{P_p \in P: f(P_p) < f(c), u(P_p) < u(c)\} = \emptyset$

2. Ranking step

   Calculate the fitness value f(*c*) and the unfitness value u(*c*) of the best found offspring.

   for $p = 1, \ldots, s_1 + s_2$

   Calculate the fitness value $f(P_p)$ and the unfitness value $u(P_p)$ of individual *p*.

   if $f(P_p) \geq f(c)$ and $u(P_p) \geq u(c)$ **then**

   $S_1 \leftarrow S_1 \cup P_p$

   else if $f(P_p) < f(c)$ and $u(P_p) \geq u(c)$ **then**

   $S_2 \leftarrow S_2 \cup P_p$

   else if $f(P_p) \geq f(c)$ and $u(P_p) < u(c)$ **then**

   $S_3 \leftarrow S_3 \cup P_p$

   else

   $S_4 \leftarrow S_4 \cup P_p$

   end if

   end for

3. Selection step

   if S_1 is not empty **then**

   Select the population member from S_1 with the worst unfitness value, with ties broken by worst fitness. Say the selected member is individual *x*, where $x \in P$.

   $P \leftarrow (P \cup c) - x$

   else if S_2 is not empty **then**

   Select the population member from S_2 with the worst unfitness value, with ties broken by worst fitness. Say the selected member is individual *x*, where $x \in P$.

   $P \leftarrow (P \cup c) - x$

   else if S_3 is not empty **then**

   Select the population member from S_3 with the worst unfitness value, with ties broken by worst fitness. Say the selected member is individual *x*, where $x \in P$.

   $P \leftarrow (P \cup c) - x$

   else

   The new offspring is not better than any of the current population members

   end if

4. Repeat the whole process of reproduction and replacement. The new population is given as an input variable for algorithm 3.3. Repeat the process until the stopping criterion is satisfied.

3.2 The Hybrid Genetic Algorithm

The process to find a near-optimal solution can be accelerated by introducing the concept of moving to neighbourhood solutions. This section is used to describe the Hybrid Genetic Algorithm. The same procedures as described in Section 3.1 are executed, but the heuristic is expanded with two processes.

The first procedure tries to reduce the unfitness value of each population member after the generation of the initial population and of each offspring. After the generation of an individual, the unfitness value associated with this member is calculated. When the unfitness value is larger than zero, the customer supplied by the first vehicle with a constraint violation is selected that causes the least increase in distance when re-assigned to its best position in the next numbered vehicle. Only when the total unfitness is reduced, this movement of one customer is maintained. The process is repeated for all vehicle with a constraint violation. After that, the same procedure is repeated, but customers are now re-assigned to its best position in the previous numbered vehicle. The process ends when the unfitness value of the individual is equal to zero, or when the unfitness value cannot be reduced any further. This procedure is used to find feasible individuals, or at least to reduce the unfitness of the individuals.

The second expansion performs two types of neighbourhood search on each population member of the initial population and after every 10,000 successful generations. First, the vehicle routes are represented as a single Travelling Salesman Problem solution, with the depot replicated between each route. For each potential move, the fitness and unfitness values are calculated and the best move is made at each iteration. This is called the 2-optimal method. When no further improvements can be done, a second neighbourhood search is performed. First, customers are removed from the route one by one and inserted into their best position in an adjacent route. Due to the numbering of the vehicle routes, an adjacent route is a route with a number that differs one from the other route. Secondly, customers are swapped between adjacent routes. Again, the best move is made at each iteration until no further improvements can be done. In that case, the individual routes must be re-optimised using the 2-optimal method.

3.3 Modifications of the Genetic Algorithm

The aim of this study is to provide a Genetic Algorithm for the Vehicle Routing Problem which is competitive with other heuristics in terms of computing time and solution quality. Besides implementing the Genetic Algorithm described by Baker and Ayechew [1], it is also important to determine whether the results can be improved. To do so, some parts of the Genetic Algorithm will be adapted. This section is used to explain all variations that were tried.

After implementing one of the modifications described in this chapter, the results must be analysed. When the quality of the results is somewhat better, but the computation time is increased extremely, this modification seems not to be a good one. The improvement in quality and the increase in computation time should be balanced.

3.3.1 Modifications of the generation of the initial population

First of all, the sweep approach of Gillett and Miller [11] to generate a population of structured solutions can be adjusted. It may be beneficial to change the conditions for including the last customer in the sweep. When the values of R_c and R_d are increased towards one, the last customer is added to the vehicle less often. In that way, individuals are created with a smaller unfitness value. It may not be possible to increase the values of R_c and R_d to one, because not all customers can be assigned to a vehicle in that way. We will check to which two-decimal number these condition values can be increased.

Half of the initial population is generated with the sweep approach, the other half of the initial population is generated with the generalised assignment approach. It is good to have much diversity between the individuals in the initial population. Diversity between the individuals in the population is an important aspect that is necessary to find near-optimal solutions. Let s_1 and s_2 be the number of individuals created with the sweep approach and with the generalised assignment approach, respectively. The summation of these two numbers equals the total number of population members that must be created, say s.

We can vary the values of s_1 and s_2, such that a proportion s_1 / s of the individuals in the initial population is generated with the sweep approach and a proportion s_2 / s of the individuals in the initial population is generated with the generalised assignment approach. Diversity within the population is required in order to support efficient search. We do not expect much improvement by varying s_1 and s_2 such that $s_1 \neq s_2$ in comparison with the results obtained when $s_1 = s_2$.

3.3.2 Modifications of the reproductive process

During the reproductive process, a binary tournament method is used to select two parent solutions from the population. In that case, the tournament size is equal to two. For every parent, two individuals are randomly chosen from the population and the one with the better fitness value is selected. A good modification may be to change the number of individuals that are chosen from the population. In general, a *k*-tournament method can be used to select parent solutions from the population. The tournament size is equal to *k* and the best individual of *k* randomly chosen population members is selected as a parent. Because two different parent solutions must be selected, *k* needs to be greater than zero and smaller than the total number of population members.

An increased tournament size will decrease the probability that an individual with a bad fitness value is selected. However, there are also disadvantages associated with the increase of the tournament size. The best adjusted individuals will be selected more often with a large tournament size, which will decrease the diversity of the population. For efficient search, it is important that there is a proper balance between genetic quality and diversity within the population. A smaller tournament size will increase the probability that an individual with a high fitness value is selected as a parent. This may result in the creation of bad offspring, but it can also increase the diversity in the population. When the tournament size is equal to one, each population member is selected to be a parent with the same chance, independent of the fitness and unfitness values of the individuals. We will test whether it is beneficial to set the tournament size equal to one and/or to three.

The performance of the Genetic Algorithm can be improved by mutating the new offspring. Baker and Ayechew [1] found that selecting two genes at random and exchange their values was the most efficient mutation of all mutations that they considered. Another possibility to mutate the offspring is the following. First, generate the random variable U, which is uniformly distributed between zero and one. When the value of U is smaller or equal to 0.5, two genes are selected at random and their values are exchanged. However, when the value of U is between 0.5 and 0.8, this step is performed two times. So two pairs of two genes are selected at random and the values of each pair of two genes are exchanged. When U is larger than 0.8, three pairs of two genes are selected at random and the same process is executed.

The values 0.5 and 0.8 requires some explanation. These values are chosen in such a way that approximately half of the newly generated offspring are exposed to the simple mutation of selecting two genes at random and exchanging their values when these are not the same. This is the kind of mutation that happens most of the time in practice. The number of offspring for which two pairs of two genes are selected is, approximately, 1.5 times the number of offspring which are exposed to the most complex mutation mentioned in this subsection.

3.3.3 Modification of the replacement process

A possible modification during the replacement phase is to insert both offspring in the population, when there is for both offspring an individual that has a higher fitness value and/or a higher unfitness value. One drawback is that the diversity between the individuals in the population will decrease over time when both parents remain in the population and both offspring are added. This may speed up the convergence process and it is likely that we end up in a local minimum.

3.3.4 Modifications with respect to the stopping criterion

Finally, the stopping criterion can be varied. The algorithm can terminate when the pre-specified number of successful generations is produced or when the solution quality is sufficient. Another possible stopping criterion is the achievement of a pre-specified, maximum computation time.

4. Data

This chapter is used to describe the data that are used for this study. Sections 4.1 describes how the data are collected and Section 4.2 is used to describe the data in much detail.

4.1 Data collection

There are many data sets available for which benchmark results are known, see for example [15]. However, the aim of this study is to implement the Genetic Algorithm described by Baker and Ayechew [1] and to verify their results. On that account, the Genetic Algorithm is applied to routing instances that were used by Baker and Ayechew [1]. The fourteen instances that are used by Baker and Ayechew can be downloaded from the OR-library, see Beasley [16].

4.2 Data description

The fourteen routing instances that are used by Baker and Ayechew [1] are different in terms of the number of customers that must be supplied and the associated locations of these customers, the capacities of the vehicles, the drop allowance associated with each customer and the distance limit of the vehicles. The smallest instances contain only fifty customers, while the demand of 200 customers must be supplied in the largest instances. The capacity per vehicle ranges from 140 to 200 goods that the customers demand. When there is a distance limit per vehicle, a drop allowance greater than zero is associated with each customer. Seven of the fourteen vehicle routing instances do not have a limit on the distance that each vehicle in the fleet can travel. However, due to the fact that each vehicle has a capacity, it is not possible that one vehicle is sufficient to fulfil all demand in one trip.

Scatter plots of two problem instances described in the paper written by Baker and Ayechew [1] are shown in Figure 4.1 and Figure 4.2. The customers in Figure 4.1 are randomly distributed around the depot, there is no pattern in their distribution. Customers who are located near each other should be supplied by the same vehicle, if possible. The customers in Figure 4.2 are grouped in clusters.

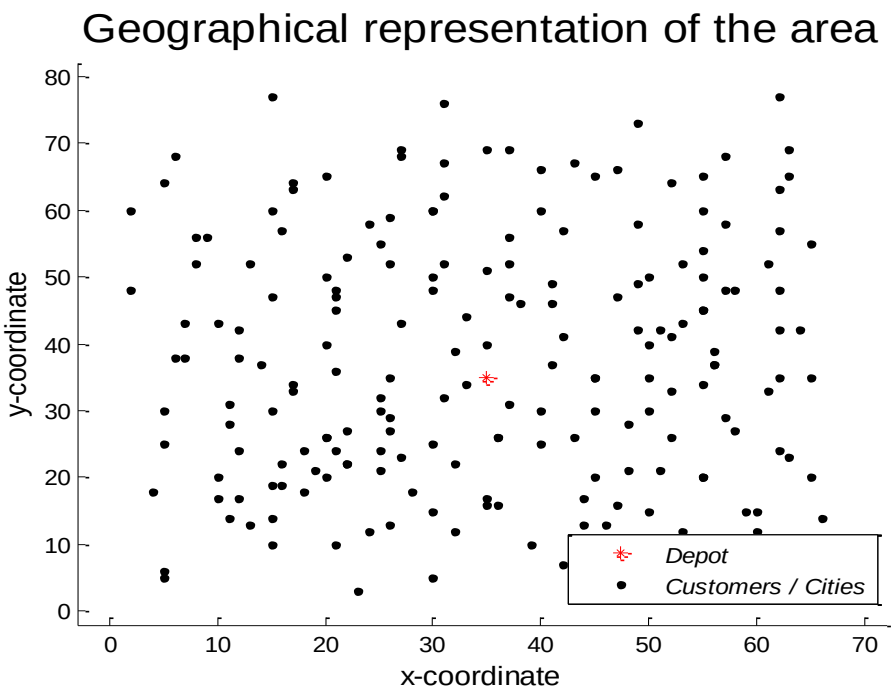


Figure 4.1: Customers are randomly distributed around the depot, instances one and six. Later in this study, these two problem instances are denoted as instance one and instance two, respectively.

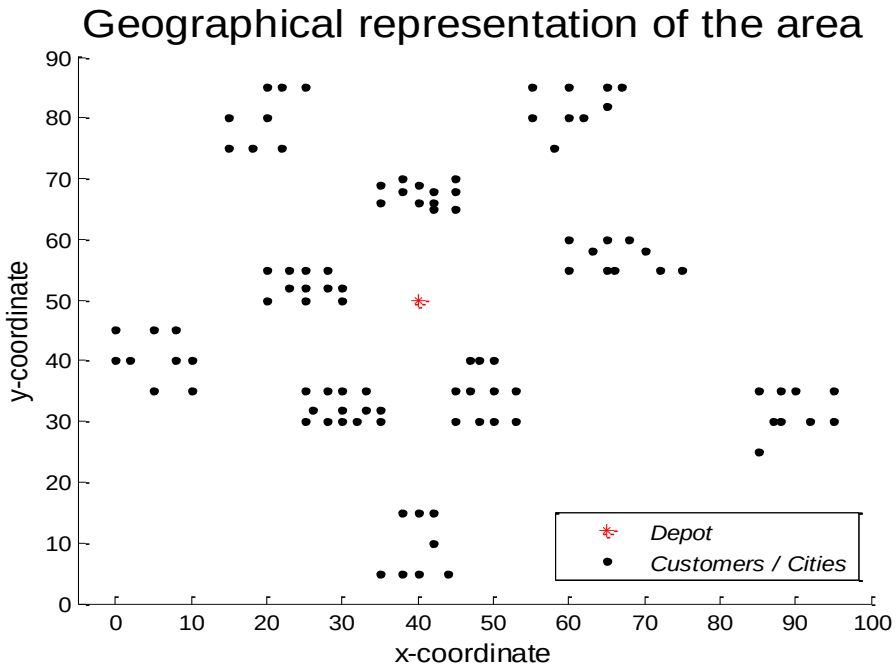


Figure 4.2: Customers are grouped in clusters, instances twelve and fourteen. Later in this study, these two problem instances are denoted as instance three and instance four, respectively.

Baker and Ayechew [1] used the programming language C to code the algorithm. Due to time limitations and the long computation time of Matlab codes in comparison with the programming language C, we decided to chose four vehicle routing instances out of the fourteen instances discussed by Baker and Ayechew [1]. The selected instances are respectively instance one, six, twelve and fourteen. Vehicle routing instances one and six are in fact the same instances, with the

customers located at the same places. The customers are randomly distributed around the depot, see Figure 4.1. The only difference between these two instances is that the vehicles in the last instance have a distance limit and each customer has an associated drop allowance. Later in this study, these two instances are denoted as instance one and instance two, respectively. The same relation holds for instance twelve and fourteen, where customers are grouped in clusters. Later in this study, these two instances are denoted as instance three and instance four, respectively. The main features of the four vehicle routing instances are represented in Table 4.1.

Table 4.1: Main features of the selected vehicle routing instances which can be downloaded from the OR-library, see Beasley [16].

Instance number	**Number of customers**	**Capacity**	**Drop allowance / distance limit**	**Number of vehicles, determined by Baker and Ayechew [1].**	**Total demand**
1	50	160	0/∞	5	777
2	50	160	10/200	6	777
3	100	200	0/∞	10	1810
4	100	200	90/1040	11	1810

As can be seen from Table 4.1, it is assumed that the drop allowance associated with each customer is the same for all customers in one problem set. As expected, the number of vehicles needed to satisfy all demand without violating the constraints is higher in the case where each vehicle has a maximum travelling distance.

5. Analysis

Section 5.1 is used to analyse the results that are obtained after applying the Genetic Algorithm to the four routing instances. The results obtained when the algorithm is carried out for two instances for which exact solutions are known, are described in Section 5.2. The last section provides an evaluation of the different modifications mentioned in Section 3.3.

5.1 Results of the Genetic Algorithm

As mentioned before, the high-level technical computing language Matlab® release R2011a is used to code the Genetic Algorithm described in Chapter 3. The computer on which the code is executed has an Intel(R) Core(TM) i5-3317U 1.70GHz processor. Baker and Ayechew [1] have coded the algorithm in C and they used a computer with an Pentium 266 MHz processor.

Our results must be compared to the results found by Baker and Ayechew [1] and to the best known results to benchmark Vehicle Routing Problems obtained by Taillard [7] and Rochat and Taillard [8]. There are two ways in which our results can differ from the results found by Baker and Ayechew [1]. On one hand, the results can differ in solution quality. However, when the algorithm described in the paper is correctly implemented, the fitness values of the best found individual should be more or less the same. The best individual in the population is the member with the lowest unfitness value, with ties broken by lowest fitness value. It is more likely that the results will differ in terms of solution time, due to the fact that the Genetic Algorithm is implemented in a different programming language and a computer with another central processing unit is used. C, the programming environment that is used by Baker and Ayechew [1], is relative fast in comparison with Matlab.

We discovered that only the value of R_d needs to reduce for the second instance. It was not possible to generate half of the initial population with the sweep approach for the second vehicle routing instance with the pre-specified value of R_d. The customers are located randomly around the depot, so the sorting method is based on the polar angle between the customers and the depot. However, this method was not sufficient for this instance. The sorting method based on the nearest neighbour solution to the Travelling Salesman Problem is used. In that way, enough unique individuals could be generated during the sweep approach.

Table 5.1 shows the best known results to benchmark Vehicle Routing Problems that were obtained with the tabu search implementations of Rochat and Taillard [8]. Also, the results obtained using simulated annealing [9] are represented. As mentioned by Baker and Ayechew [1], there is variability of the Genetic Algorithms' performance from one run to another. To that extend, they carried out six independent runs of the Pure Genetic Algorithm for the four routing instances. The best, worst and average solution values over six independent runs of the Pure Genetic Algorithm and the solution values of the Hybrid Genetic Algorithm found by Baker and Ayechew [1] are included in Table 5.1.

Table 5.1: Comparison of the Genetic Algorithm with published results. The results of the Genetic Algorithm stated in this table are the ones found by Baker and Ayechew [1]. This results are obtained with 3-exchange during the travelling salesman heuristic. The best, worst and average result for the Pure Genetic Algorithm are included to show the performance of this algorithm over six independent runs. The values represent the total distance travelled by all vehicles of the individual, excluding any drop allowance.

Instance number	Osman[1] (SA)	Rochat and Taillard[2] (TS)	Pure GA[3] (Best)	% Gap	Pure GA[3] (Worst)	% Gap	Pure GA[3] (Average)	Hybrid GA[4]	% Gap
1	528	524.61	524.81	0.04	524.81	0.04	524.81	524.61	0.00
2	555	555.43	559.04	0.65	560.29	0.87	560.08	555.43	0.00
3	826	819.56	863.73	5.39	878.37	7.18	871.38	819.56	0.00
4	890	866.37	872.34	0.69	891.15	2.86	887.70	867.13	0.09

After testing our Genetic Algorithm for small problem instances, the algorithm is carried out for each of the four instances. Due to the fact that the number of generations for the Pure Genetic Algorithm is 100,000 for the first two instances and 200,000 for the last two instances, the computing time before termination is long. To reduce the amount of time needed to find good vehicle routes for the instances, it seems to be a good idea to simplify the travelling salesman procedure by leaving out the 3-exchange phase. However, this is only advisable when the solution quality does not decrease much. We provide the results obtained by using a travelling salesman heuristic with 3-optimal refinement and the results obtained by using a travelling salesman heuristic that only includes a greedy algorithm followed by the 2-optimal heuristic.

[1] Simulated annealing by Osman [9].

[2] Tabu search by Taillard [7] and by Rachot and Taillard [8].

[3] Solution obtained by Pure Genetic Algorithm and percentage above best published result (by Rochat and Taillard [8]).

[4] Best solution obtained by Hybrid Genetic Algorithm with neighbourhood search and percentage above best published result (by Rochat and Taillard [8]).

The results obtained by carrying out the Genetic Algorithm without the 3-optimal heuristic for each of the problem instances and the results obtained by using the Genetic Algorithm with the 3-optimal heuristic are represented in Table 5.2 and Table 5.3, respectively. The best, worst and average result over four independent runs of the Pure Genetic Algorithm are included to show the performance of this algorithm. When the 3-optimal heuristic is discarded, only pair wise exchange is used to improve the tours. The fitness value of the best individual after creating the initial population during the best run is represented in the second column. The best run is one of the four independent runs for which the final best individual has the lowest fitness. It was observed that the initial population did not include a feasible solution for the second problem instance when the Pure Genetic Algorithm without 3-optimal refinement is used. The unfitness value of the best individual in the initial population is represented between brackets.

Table 5.2: Results of the Genetic Algorithm. This results are obtained without the 3-optimal refinement during the travelling salesman heuristic. The best, worst and average results for the Pure Genetic Algorithm are included to show the performance of this algorithm over four independent runs.

Instance number	Pure GA (Initial, Best)	Pure GA[3] (Best)	% Gap	Pure GA[3] (Worst)	% Gap	Pure GA (Average)	Hybrid GA[4] (Final)	% Gap
1	532.70	524.63	0.00	524.63	0.00	524.63	524.61	0.00
2	658.59 (0.01)	556.68	0.23	562.65	1.30	558.58	560.89	0.98
3	947.29	825.65	0.74	877.36	7.05	846.38	825.95	0.78
4	1114.65	881.11	1.70	910.36	5.08	890.27	866.37	0.00

Figure 5.3: Results of the Genetic Algorithm. This results are obtained with the 3-optimal refinement during the travelling salesman heuristic. The best, worst and average results for the Pure Genetic Algorithm are included to show the performance of this algorithm over four independent runs.

Instance number	Pure GA (Initial, Best)	Pure GA[3] (Best)	% Gap	Pure GA[3] (Worst)	% Gap	Pure GA (Average)	Hybrid GA[6] (Final)	% Gap
1	531.90	524.61	0.00	524.61	0.00	524.61	524.61	0.00
2	595.87	556.68	0.23	560.29	0.87	557.94	560.89	0.98
3	956.97	825.95	0.78	866.34	5.71	846.36	824.78	0.64
4	1106.89	888.01	2.50	972.60	12.26	921.55	866.37	0.00

The following three paragraphs are used to evaluate the results stated in Table 5.2. We were not able to find an individual in the initial population with a fitness value of zero for the second vehicle routing instance. We adjusted the sweep approach for that instance. The generalised assignment approach creates almost always individuals with an unfitness value larger than zero. By adjusting the sweep approach in such a way that the last customer in the sweep is added more often, larger

constraint violations will occur. This may be the explanation for the fact that there are no feasible individuals in the initial population for the second vehicle routing instance.

The final results found with the Pure Genetic Algorithm do not differ very much from the results found by Baker and Ayechew [1]. It is remarkable that we found a lower fitness value in three of the four cases, while the heuristic to make an explicit route is simplified by leaving out the 3-optimal refinement heuristic. The average fitness value of the best feasible solutions found in our study is lower than the average fitness value of the best found feasible solutions found by Baker and Ayechew [1] for the first three instances. There is more volatility in the performance of the Pure Genetic Algorithm for the last two instances, which coincides with the observations made by Baker and Ayechew [1].

The two tables below show the tour of the best found individual for the second vehicle routing instance with the tabu search implementation by Rochat and Taillard [8] and the one we found with the Pure Genetic Algorithm without 3-optimal refinement. As can be seen from Table 5.4 and Table 5.5, three pair of routes serve exactly the same customers in the same order. The fitness value of the best found solution with the tabu search implementation of Rochat and Taillard [8] is 1.25 lower than the one found with our Genetic Algorithm. For the other vehicle routing instances, these tables can be found in Appendix A1 - A4.

Table 5.4: Tour of the best found individual for the second instance by Rochat and Taillard [8]. This data can be downloaded from the OR-library, see Beasley [16].

Vehicle number	**Fitness value**	**Total demand**	**Route of the different vehicles in the fleet. The depot is represented by 0.**											
1	99.12	155.00	0	5	49	10	39	33	45	15	44	37	12	0
2	108.08	137.00	0	1	22	31	28	3	36	35	20	2	0	
3	109.94	131.00	0	17	42	19	40	41	13	25	14	0		
4	42.33	80.00	0	46	47	4	18	0						
5	100.64	133.00	0	27	48	8	26	7	43	24	23	6	0	
6	95.33	141.00	0	38	9	30	34	50	21	29	16	11	32	0
Total:	555.43	777.00												

Table 5.5: Tour of the best found individual associated with the best fitness value represented in Table 5.2 for the second vehicle routing instance, found with the pure Genetic Algorithm without the 3-optimal refinement phase during the travelling salesman procedure.

Vehicle number	**Fitness value**	**Total demand**	**Route of the different vehicles in the fleet. The depot is represented by 0.**											
1	108.08	137.00	0	1	22	31	28	3	36	35	20	2	0	
2	100.64	133.00	0	6	23	24	43	7	26	8	48	27	0	
3	50.46	90.00	0	14	25	18	0							
4	109.07	132.00	0	17	44	42	19	40	41	13	4	47	0	
5	93.10	144.00	0	12	37	15	45	33	39	10	49	5	46	0
6	95.33	141.00	0	32	11	16	29	21	50	34	30	9	38	0
Total:	556.68	777.00												

Fortunately, the results obtained without the 3-exchange procedure are comparable to the results obtained with the 3-exchange step. This can be concluded after comparing Table 5.2 and Table 5.3. The best found individual over four independent runs carried out with the Pure Genetic Algorithm without the 3-optimal refinement incorporated in the travelling salesman heuristic has a lower fitness value than the individual found by using the Pure Genetic Algorithm with 3-exchange for the last two instances. Also the results obtained by using the Genetic Algorithm Hybridised with neighbourhood search are highly comparable. The CPU times in seconds are represented in Table 5.6.

The computing time decreases significantly when the explicit tours are created with only pair wise exchange. For the Hybrid Genetic Algorithm, incorporating the neighbourhood search techniques described in Section 3.2, the integer between brackets represents the number of successful generations after which the algorithm terminates, for the best run. The Genetic Algorithm Hybridised with neighbourhood search terminates when there is no improvement in 20,000 successive runs.

As can be seen from Table 5.6, the computing times are long in comparison with the computation times stated in the article of Baker and Ayechew [1]. There are some explanations for this observation. First of all, we coded the heuristic in the high-level technical computing language Matlab® release R2011a. Matlab is an interpreted language which is usually slower than compiled languages as C, the programming language that is used by Baker and Ayechew [1]. Nested loops slows down the process in Matlab extremely. Secondly, the computer that we have used has another processor than the one used by Baker and Ayechew [1]. The travelling salesman heuristic, also without the 3-exchange phase, requires a lot of computing time. This is mainly caused by the indeclinable fact that the process to make explicit routes is called very often.

Table 5.6: Comparison of CPU times (in seconds) with published results by Baker and Ayechew [1].

Instance number	Gendreau et al[5] (TABU-ROUTE)	Pure GA, Baker and Ayechew [1][6]	Pure GA, without 3-exchange[7]	Pure GA, with 3-exchange[8]	Hybrid GA, Baker and Ayechew [1][9]	Hybrid GA, without 3-echange[10]	Hybrid GA, with 3-exchange[11]
1	360	213	5,314	14,692	23	4,001 (30,000)	3,785 (20,002)
2	810	217	12,174	24,117	429	2,991 (26,222)	7,198 (24,564)
3	960	1,160	18,713	40,372	1,285	5,998 (21,844)	8,647 (20,700)
4	3,942	1,197	22,963	48,623	585	8,399 (37,722)	15,129 (30,000)

The Hybrid Genetic Algorithm without 3-exchange carried out on the first vehicle routing instance takes more time than the Hybrid Genetic Algorithm with 3-exchange. The fact that the neighbourhood search is executed four times during the first algorithm and only three times during the second one may be an explanation. For the other three instances, the neighbourhood search function is invoked equally often for both variations of the Hybrid Genetic Algorithm.

5.1.1. Methods to reduce the computing time

The number of successful generations after which the Pure Genetic Algorithm terminates is 100,000 for instances with at most fifty customers and 200,000 for vehicle routing instances which incorporate more customers. By a successful generation we mean that an offspring is found that does not duplicate an existing population member. The computing time can be reduced by decreasing the number of successful generations before termination. However, such a reduction is only desirable when the best individual found is created in an early generation. In this subsection, we discuss the results associated with the lowest fitness value found over four independent runs of the Pure Genetic Algorithm without 3-optimal refinement in the travelling salesman heuristic, for each of

[5] Tabu search by Gendreau et al., seconds using Silicon Graphics work station (5.7 Mflow/s)
[6] Pure Genetic Algorithm, seconds using Pentium 266 MHz.
[7] Pure Genetic Algorithm without 3-exchange during travelling salesman heuristic, seconds using Intel Core 1.70GHz.
[8] Pure Genetic Algorithm with 3-exchange during travelling salesman heuristic, seconds using Intel Core 1.70GHz.
[9] Genetic Algorithm Hybridised with neighbourhood search, seconds using Pentium 266 MHz.
[10] Genetic Algorithm Hybridised without 3-exchange during travelling salesman heuristic, seconds using Intel Core 1.70GHz.
[11] Genetic Algorithm Hybridised with 3-exchange during travelling salesman heuristic, seconds using Intel Core 1.70GHz.

the four instances. Figure 5.1 shows the convergence of the fitness value of the best found individual with the Pure Genetic Algorithm without 3-exchange. The fitness value of the best individual is calculated every fifty generations. In that way, we get an adequate representation of the best found fitness value and the computing time is not increased much.

From Figure 5.1 we can conclude that the number of successful generations before termination can be reduced significantly, especially for the first three instances. One remark should be made here. As can be seen from the figure for the second vehicle routing instance, the initial fitness value decreases until generation 500. After that, it increases slightly and it starts decreasing around generation 700. Due to the fact that this figure represent the fitness value of the best found individual, this means that there was no individual with an unfitness value of zero until generation 700. We saw earlier that there was no individual with an unfitness value of zero in the initial population for this instance, which coincides with the observation just made. An individual is better than another individual when its unfitness value is lower than the unfitness value of the other individual. All final best found individuals have an unfitness value of zero.

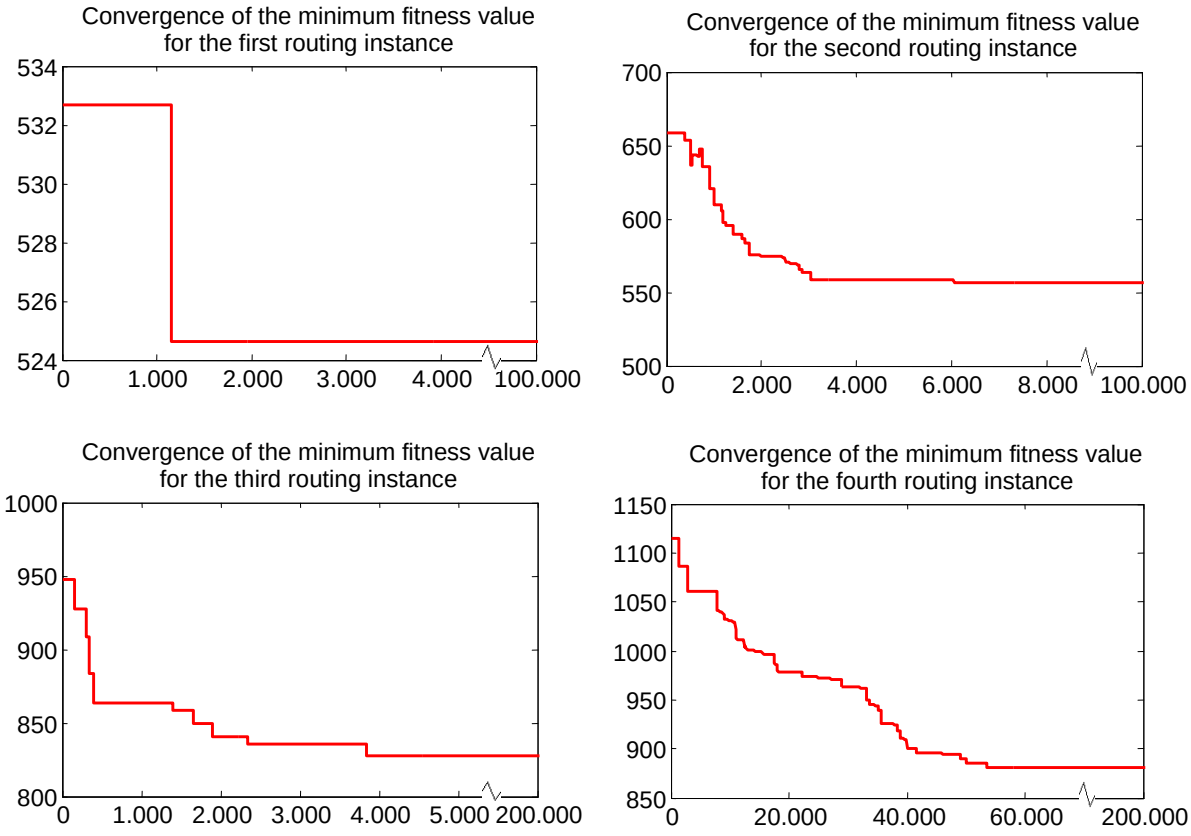


Figure 5.1: Convergence of the fitness value of the best individual in the population, for each of the four vehicle routing instances.

Individuals whose vehicles serve exactly the same customers are called duplicating individuals, irrespective of the order in which the customers are supplied. An offspring that duplicates an existing member of the population is discarded. In that case, new offspring must be created. The number of successful generations is only updated when at least one unique offspring is created. Figure 5.2 shows the number of times that a specific number of trials is performed. Duplications are checked for after the creation of the offspring, applying the mutation and the check whether or not all vehicles in the fleet are used. In the case that only one offspring duplicates an existing member, the other offspring is selected to be the best offspring. When both offspring duplicates a population member, new offspring must be created. The individual with the lowest unfitness value, with ties broken by lowest fitness, is selected when both offspring are unique with respect to the current population. Whether or not the selected offspring replaces one of the population members depends on the fitness value and unfitness value of both the population members and the best unique offspring.

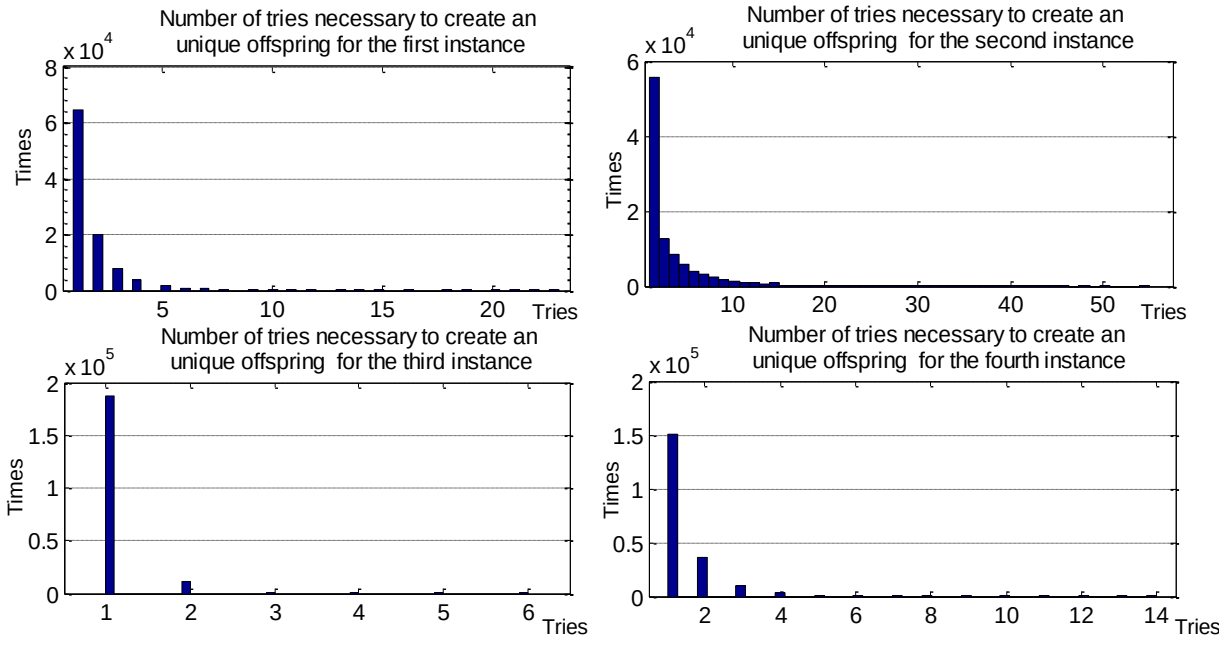


Figure 5.2: Number of tries before at least one unique offspring is created.

There is no pattern in the number of tries before at least one unique offspring is created. There is enough diversity within the population to create unique offspring, also in the last phase of the algorithm where the individuals are converged. The number of generations is actually much higher than the pre-specified number of 100,000 for the two smallest instances and 200,000 for the last two vehicle routing instances. The precise numbers are represented in Table 5.7. As mentioned before, the results discussed in this subsection are the once associated with the lowest fitness value found over four independent runs of the Pure Genetic Algorithm without 3-optimal refinement in the travelling salesman heuristic, for each of the four routing instances.

Table 5.7: Actual number of generations and the number of successful generations after which the pre-specified number of generations is achieved, for each of the four vehicle routing instances.

Instance number	Pre-specified number of generation	Actual number of generations	Number of successful generations after which the pre-specified number of generations is produced
1	100,000	169,393	62,052
2	100,000	341,209	30,226
3	200,000	213,128	187,681
4	200,000	270,788	147,297

An unsuccessful generation refers to the generation of duplication offspring, which are discarded. New parents must be selected, whose genes are used to create offspring. This process is repeated until at least one of the offspring is unique with respect to the population members, after applying a simple mutation to the new offspring. The most important conclusion that we can draw from Table 5.7 is that the number of unsuccessful generations is remarkable for all four routing instances, e.g. the number of unsuccessful tries for the first instance is 69,393 (= 169,393 – 100,000). As reported in the table, the pre-specified number of generation - successful and unsuccessful - is reached after 62,052 successful generations. However, we interpreted the pre-specified number of generations as the number of successful generation that must have been occurred before the algorithm must terminate.

There are at least three modifications of the Genetic Algorithm that will reduce the computing time remarkably without making the solution quality worse. First, we can reduce the number of successful generations after which the algorithm should terminate. As can be seen from Figure 5.1, the number of successful generations after which the algorithm must terminate can be reduced quite a lot for all four instances. For the first and third vehicle routing instances, a pre-specified number of 10,000 successful generations is save. That number is significantly smaller than the pre-specified number of 100,000 and 200,000 for the first and third instances, respectively. An adequate number of generations for the second instance lies around 15,000 and the Pure Genetic Algorithm applied to the fourth instance can terminate after 100,000 successful generations.

On the other hand, we could also call every creation of new offspring a successful generation, irrespective of the fact whether or not both children duplicate an existing population members. Of course, offspring which duplicate existing members of the population are discarded. In that case, the number of times an unique offspring is found lies around the values represented in the last column of Table 5.7. As mentioned before, there is some volatility from one run to another.

Another option is to terminate whenever there is no improvement in, say, 20,000 successive generations. Comparing Figure 5.1 with Table 5.7 leads to the conclusion the first method will reduce the computing time the most.

5.2 Performance of the algorithm for instances for which exact solutions are known

Two vehicle routing instances that can be solved exactly with the flow formulations described in Section 2.2 are used to evaluate whether the Genetic Algorithm is adequate to solve instances exactly. The selected instances can be downloaded from SYMPHONY VRP Instances, see Ralphs [17]. More information about these routing instances, like the computational results to find the exact solution, can be found in the paper of Augerat et al. [18]. Table 5.8 shows the most important features of the two vehicle routing instances.

Table 5.8: Main characteristics of the selected vehicle routing instances which can be downloaded from SYMPHONY VRP Instances, see Ralphs [17].

Instance name	Number of customers	Capacity	Drop allowance / distance limit	Number of vehicles	Total demand	Optimal fitness value
A-n32-k5	31	100	0/∞	5	410	784
A-n48-k7	47	100	0/∞	7	626	1073

We have also calculated the fitness values associated with the optimal tours found by Augerat et al. [18], to check whether the results are correct. However, the calculated fitness values are different than the ones stated in Table 5.8. The fitness value associated with the tour found for instance A-n32-k5 is 787.81 and the one associated with the tour found for instance A-n48-k7 is 1074.34. The results obtained with the Genetic Algorithm are compared with these fitness values.

The customers are randomly distributed around the depot in both instances. Instance A-n32-k5 has only 31 customers with known demand. The number of population members is set equal to twenty for this vehicle routing instance. The initial population is fully generated with the standard sweep approach, because no individuals can be created with the generalised assignment approach. Instance A-n48-k7 has 47 customers. Fifteen members of the initial population are generated with the sweep approach and the other fifteen members are generated with the generalised assignment approach.

The results found without 3-optimal refinement during the travelling salesman heuristic and the results found with the 3-optimal heuristic are represented in Table 5.9 and Table 5.10, respectively. The percentage between the total distance travelled by all vehicle in the optimal solution and the total distance travelled by all vehicles in the solutions we found is also displayed. A negative value means that we found a tour that is better than the 'optimal' tour.

Table 5.9: Comparison of the Genetic Algorithm with exact results. These results are obtained without 3-exchange during the travelling salesman heuristic. The values represent the total distance travelled by all vehicles of the individual, excluding any drop allowance.

Instance number	Pure GA (Initial)	Pure GA[3] (Final)	% Gap	Hybrid GA[4] (Final)	% Gap
A-n32-k5	885.04	789.06	0.16	787.08	-0.09
A-n48-k7	1205.68	1074.34	0.00	1074.34	0.00

Table 5.10: Comparison of the Genetic Algorithm with exact results. These results are obtained with 3-exchange during the travelling salesman heuristic. The values represent the total distance travelled by all vehicles of the individual, excluding any drop allowance.

Instance number	Pure GA (Initial)	Pure GA[3] (Final)	% Gap	Hybrid GA[4] (Final)	% Gap
A-n32-k5	885.04	787.08	-0.09	797.45	1.22
A-n48-k7	1178.94	1074.34	0.00	1074.34	0.00

Again, the results obtained without the 3-exchange procedure are comparable to the ones obtained with the 3-exchange step. This can be concluded after comparing Table 5.9 and Table 5.10. We discuss the results stated in Table 5.9. The fitness value of the best individual after creating the initial population is represented in the second column. The final results found with the Pure Genetic Algorithm do not differ very much from the exact results found by Augerat et al. [18]. The result obtained with the Hybrid Genetic Algorithm is better than the result obtained with the Pure Genetic Algorithm for instance A-n32-k5. Our best found individual has a fitness value lower than the fitness value of the optimal tour found by Augerat et al. [18]. The individuals found for instance A-n48-k7 with the Pure Genetic Algorithm and the Genetic Algorithm Hybridised with neighbourhood search are exactly the same as the one found by Augerat et al. [18].

Appendix B1-B2 contains the optimal tour found by Augerat et al. [18] and the tours of the best individuals that were found with the Pure Genetic Algorithm without 3-optimal refinement and the Hybrid Genetic Algorithm without 3-optimal refinement during the process to make the routes explicit, for both instances

.

5.3 Results of the modifications

This section is used to describe the results that are obtained after adjusting the Genetic Algorithm with the different modifications described in Section 3.3. Each modification is tested individually. As mentioned before, there are far more variations that may reduce the computing time or that increase the solution quality. Ideas for future research are provided in Chapter 6.

5.3.1 Results of the modifications of the generation of the initial population

This subsection is used to analyse the variations that were tried on the method of population generation. These modifications are discussed in Subsection 3.3.1.

5.3.1.1 Adjust the sweep approach

As mentioned before, the standard sweep approach implemented by Baker and Ayechew [1] could not be used to generate half of the initial population for the second vehicle routing instance. The conditions to insert the last customer in the sweep are changed. We tried to change the conditions to include the last customer in the sweep for the other three vehicle routing instances. After 20,000 successive generations, there was no improvement in the solution quality, both when the values for R_c and R_d are increased and decreased. There were indeed less individuals with an unfitness value of zero in the initial populations when the values of R_c and R_d are reduced. We want the lowest constraint violation possible, which can be achieved by increasing the values of R_c and R_d as much as possible.

The first vehicle routing instance is a tight problem, which means that most of the vehicle capacity must be filled to satisfy all demand with the specified number of vehicles. This instance is an example of a classical vehicle routing instance, where vehicles do not have a maximum travelling distance. R_c is the relevant variable. It is possible to increase the value of R_c from 0.75 to 0.76. An explanation for the fact that this value cannot be increased to 1.0 is the fact that this problem is tight. It is likely that some constraint violations are necessary initially. The third instance is not tight. The value of R_c can be increased to 1.0, which means that all individuals created with the sweep approach for this instance have an unfitness value of zero. The same conclusion holds for the last vehicle routing instance, where the value of R_d can be increased to 1.0. A value of R_d equal to 1.0 means that there is no excess distance on a vehicle. However, there can be excess capacity on a vehicle which results in an unfitness value larger than zero. We found out that all population members in the initial population have an unfitness value larger than zero when R_d is equal to 1.0, with unfitness values close to zero for individuals created with the sweep approach and unfitness values between two and four for the other individuals.

5.3.1.2 Varying the values of s_1 and s_2

Half of the initial population for the Genetic Algorithm is generated with the sweep approach, the other half of the initial population is generated with the generalised assignment approach. As already mentioned in Section 3.3, diversity between the individuals in the population is important to obtain near-optimal solutions. We checked whether it is beneficial to vary the values of s_1 and s_2, such that a proportion s_1 / s of the individuals in the initial population is generated with the sweep approach and a proportion s_2 / s of the individuals in the initial population is generated with the generalised assignment approach. s is the total number of individuals in the population.

When all individuals are generated with the same approach, there is less diversity between the individuals. This may lead to the inability to generate good offspring. Generating all individuals with the generalised assignment approach leads to more diversity in the initial population than generating all individuals with the sweep approach, which coincide with the fact that the generalised assignment approach can create far more different chromosomes than the sweep approach. However, the best individual in the initial population when the sweep approach is used has an unfitness value of zero and a lower fitness value than the best individual in the initial population when the generalised assignment approach is used, for the first, third and fourth vehicle routing instance. In the case that all individual are generated with the generalised assignment approach, the unfitness value of the best individual is larger than zero.

We discovered that there was no overall benefit from generating all individuals in the initial population with the sweep approach or with the generalised assignment approach. These findings coincide with our expectation that diversity within the population is required in order to support efficient search. Table 5.11 shows the best results obtained by carrying out two independent runs for different values of s_1 and s. The algorithm stops after 10,000 successive successful generations without improvement.

Table 5.11: Fitness value of the best found individual over two independent runs. These results are obtained with the pure Genetic Algorithm without 3-exchange during the travelling salesman heuristic. The values represent the total distance travelled by all vehicles of the individual, excluding any drop allowance.

Instance number	$s_1 = 0$ $s_2 = s$	$s_1 = ¼\ s$ $s_2 = ¾\ s$	$s_1 = ½\ s$ $s_2 = ½\ s$	$s_1 = ¾\ s$ $s_2 = ¼\ s$	$s_1 = s$ $s_2 = 0$
1	551.04	524.61	524.63	-	-
2	556.68	556.68	556.68	556.68	556.68
3	886.72	819.56	825.65	829.24	836.48
4	1007.24	970.33	881.11	1022.04	-

It is not possible to generate seventy-five or more percent of the initial population with the sweep approach for the first vehicle routing instance. Not all individuals can be created with the sweep approach for the fourth instance. As can be seen from Table 5.11, there is no combination of s_1 and s_2 tested for which the best found fitness value decreases for all vehicle routing instance with reference to $s_1 = s_2$. Besides saying that $s_1 = s_2$ is a good option, another good option seems to be s_1 = ¼ s and s_2 = ¾ s. In that case however, the fitness value found for the fourth vehicle routing instance is much worse than the one found when s_1 equals s_2.

5.3.2 Results of the modifications of the reproductive process

This subsection is used to analyse the variations that were tried on the method of reproduction. These modifications are discussed in Subsection 3.3.2. We have applied the modifications to the third instance. This vehicle routing instance is chosen because we were not able to find an individual with a fitness value equal to the best known result to benchmark Vehicle Routing Problem obtained with the tabu search implementations of Rochat and Taillard [8], neither with the Pure Genetic Algorithm nor the Hybrid Genetic Algorithm.

5.3.2.1 Tournament size variations

As described in Chapter 3, the binary tournament method is used to select two parent solutions from the population. In that case, the tournament size is equal to two. Variations on the tournament method were tried.

The Pure Genetic Algorithm with a tournament size of one is carried out three times for the third instance, to give an indication of the variability of the performance of the Genetic Algorithm from one run to another. The results are shown in Table 5.12.

Table 5.12: Performance of the Pure Genetic Algorithm over three independent runs. The results are obtained without 3-exchange during the travelling salesman heuristic and with a tournament size of one.

Instance number	Best	% Gap	Worst	% Gap	Average
3	825.65	0.74	829.13	1.17	826.91

When we compare Table 5.12 with Table 5.2, it is easy to see that the fitness values found with a tournament size of one are on average smaller than the ones found with a tournament size of two. There is less variation between the fitness values found in each run when the tournament size is one. However, we only carried out three independent runs. To say something about the significance of this observation, more runs should be carried out on different vehicle routing instances. The best

found individual has in both cases a fitness value of 825.65, 0.74 percent higher than the best found benchmark result.

Figure 5.3 shows the convergence of the fitness value of the best individual for runs performed with the Pure Genetic Algorithm without 3-exchange during the creation of the explicit tours and with a tournament size equal to one. As mentioned in Subsection 5.1.1, the fitness value of the best individual is calculated every fifty generations. The number of generations before the algorithm terminates is 218,735 for the run in which the best found individual has a fitness value of 825.65. This number does not differ much from the one stated in Table 5.7.

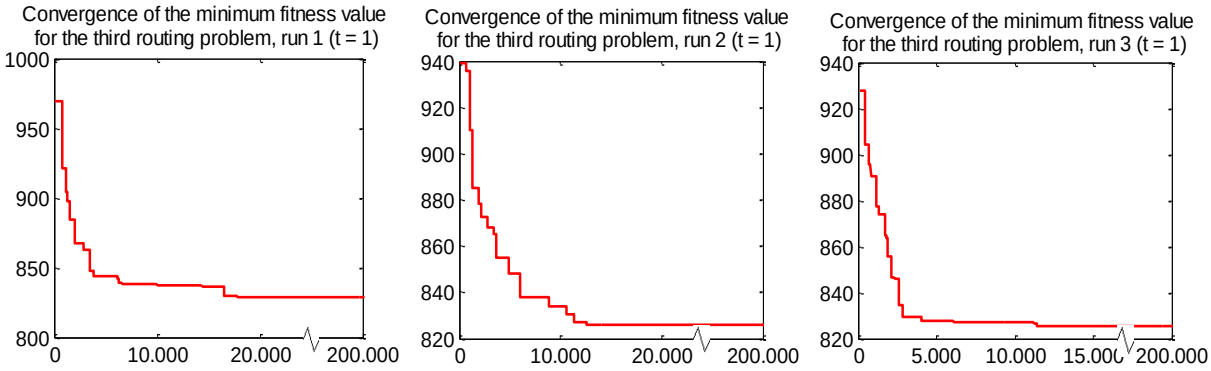


Figure 5.3: Convergence of the fitness value of the best individual for three independent runs of the Genetic Algorithm for the third vehicle routing instance with a tournament size of one.

The Pure Genetic Algorithm with a tournament size of three is also carried out three times for third instance, to give an indication of the variability of the performance of the Genetic Algorithm from one run to another. The results are shown in Table 5.13.

Table 5.13: Performance of the Pure Genetic Algorithm over three independent runs. The results are obtained without 3-exchange during the travelling salesman heuristic and with a tournament size of three.

Instance number	Best	% Gap	Worst	% Gap	Average
3	824.78	0.64	872.35	6.44	840.93

When we compare Table 5.13 with Table 5.2, it is easy to see that the fitness values found with a tournament size of three are on average smaller than the ones found with a tournament size of two. There is again quite some volatility over the different runs. The best found individual when the Pure Genetic Algorithm is performed with a tournament size of three has a fitness value which is 0.11 percent lower than the one found with a tournament size of two, whose fitness value is 825.65.

Figure 5.4 shows the convergence of the fitness value of the best individual for runs performed with the Pure Genetic Algorithm without 3-exchange during the creation of the explicit tours and with a tournament size equal to three. The number of generations before the algorithm

terminates is 213,006 for the run in which the best found individual has a fitness value of 824.78. This number does not differ much from the one stated in Table 5.7.

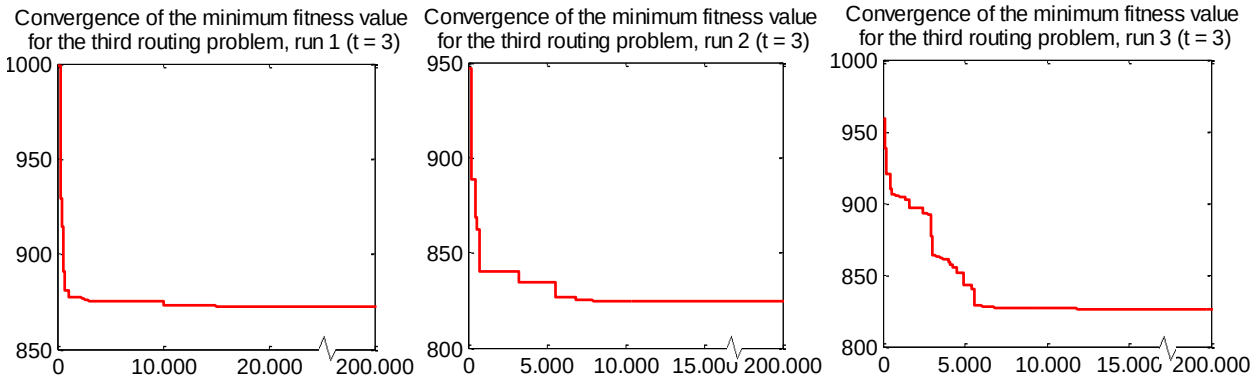


Figure 5.4: Convergence of the fitness value of the best individual for three independent runs of the Genetic Algorithm for the third vehicle routing instance with a tournament size of three.

As can be seen from Table 5.2 and Table 5.3, the best results for the third vehicle routing instance are obtained with the Hybrid Genetic Algorithm with 3-optimal refinement during the travelling salesman procedure. To check whether or not we can obtain better results than the ones found so far, we also carried out three independent runs of this algorithm with a tournament size of one. The results are shown in Table 5.14. We chose to use a tournament size of one, because the average fitness value of the best found individual is low in that case and there is less volatility than in the case when the tournament size is three. As mentioned before, more runs must be carried out to validate the observation.

Table 5.14: Performance of the Hybrid Genetic Algorithm with 3-optimal refinement over three independent runs. The results are obtained with a tournament size of one.

Instance number	Best	% Gap	Worst	% Gap	Average
3	819.56	0.00	824.78	0.64	821.30

As can be seen from Table 5.14, we were now able to find an individual with a fitness value equal to the best found benchmark result. The convergence of the fitness value of the best found candidate solution for that run is represented in Figure 5.5. The number of successful generations after which the algorithm terminates is 50,000. By that time, a total of 436 parent pairs are unsuccessfully combined.

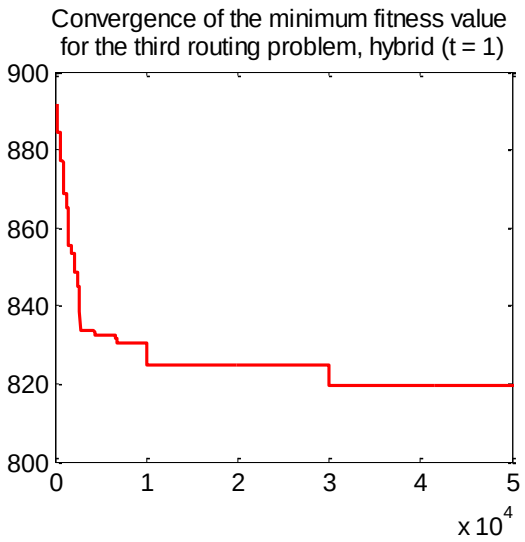


Figure 5.5: Convergence of fitness value of the best individual with the Hybrid Genetic Algorithm for the third vehicle routing instance with a tournament size of one.

5.3.2.2 Mutation method

As Baker and Ayechew [1] already mentioned in their paper, the performance of the Genetic Algorithm can be improved by mutating the new offspring. We implemented the mutation described in Subsection 3.3.2. The results of three independent runs of the Pure Genetic Algorithm with this mutation method are stated in Table 5.15. This modification does not improve the performance of the algorithm, it turns out to be non-adequate.

Table 5.15: Performance of the Pure Genetic Algorithm over three independent runs. The results are obtained without 3-exchange during the travelling salesman heuristic and with the mutation method described Subsection 3.3.2.

Instance number	Best	% Gap	Worst	% Gap	Average
3	828.26	1.06	865.00	5.54	847.57

5.3.3 Results of the modifications of the replacement process

This subsection is used to analyse the variations that were tried on the method of population generation. These modifications are discussed in Subsection 3.3.3. We have applied the modifications to the third vehicle routing instance. This vehicle routing instance is chosen because we were not able to find an individual with a fitness value equal to the best known result to benchmark Vehicle Routing Problem obtained with the tabu search implementations of Rochat and Taillard [8], neither with the pure Genetic Algorithm nor the Hybrid Genetic Algorithm.

5.3.3.1 Add all unique offspring to the population

A possible modification during the replacement phase is to insert both offspring in the population, when there is for both offspring an individual that has a higher fitness value and/or a higher unfitness value. The best offspring, the one with the lowest unfitness value with ties broken by lowest fitness, is added first. Due to the fact that in the best case both offspring are unique, the number of successful generations is now higher than in the case where at most one child is added to the population. Table 5.16 shows the results of this modification.

Table 5.16: Performance of the Pure Genetic Algorithm over three independent runs. The results are obtained without 3-exchange during the travelling salesman heuristic. Both offspring are added to the population when they are unique and have a lower fitness value and/or a lower unfitness value than one of the population members.

Instance number	Best	% Gap	Worst	% Gap	Average
3	823.04	0.42	846.05	3.23	831.58

This modification founds an individual with a fitness value lower than the best individual found with the Pure Genetic Algorithm with a tournament size of one, described in Subsection 5.3.2.1. There is more diversity between the best found individuals here. Given the possibility that both offspring are added to the population, the diversity of the population will decrease. The two offspring have one part of their chromosome from parent one and another part of the chromosome from the other parent. The only difference may be the mutation applied to new offspring, in which the values of two randomly selected genes are exchanged. When both parents remain in the population during the replacement procedure, the diversity of the population decreases. This phenomenon is also found here, as the number of generations necessary to execute lies around 280,000. Sometimes, ten generations are necessary to find at least one unique offspring. It is likely that the algorithm ends in a local minimum.

5.3.4 Best found modification

Preliminary experiments showed that decreasing the tournament size to one gives the best results on average, together with the more volatile modification of adding all unique offspring to the population. This subsection is used to discuss the results that were obtained for the other three vehicle routing instances when the tournament size is set to one, because we only showed the results obtained for the third instance in Subsection 4.3.2.1. The results obtained when three independent runs of the Hybrid Genetic Algorithm are carried out for the four instances when all unique offspring can be added to the population are discussed in this subsection.

The Pure Genetic Algorithm was, for all vehicle routing instances, terminated whenever there was no improvement in 30,000 successive generations, thereby enabling a reduction in the average computing time. As mentioned earlier, the Hybrid Genetic Algorithm was terminated whenever there was no improvement in 20,000 successive generations. The results obtained with the Pure Genetic Algorithm and the results obtained using the Hybrid Genetic Algorithm are represented in Table 5.17 and Table 5.18, respectively. The tournament size is one

Table 5.17: Performance of the Pure Genetic Algorithm over three independent runs. The results are obtained without 3-exchange during the travelling salesman heuristic and with a tournament size of one.

Instance number	Best	% Gap	Worst	% Gap	Average	Average CPU time
1	524.63	0.00	524.63	0.00	524.63	5,266.70
2	556.68	0.23	560.24	0.87	559.05	9,236.26
3	825.65	0.74	829.13	1.17	826.91	9,909.03
4	973.40	12.35	983.04	13.47	977.52	12,405.13

Table 5.18: Performance of the Hybrid Genetic Algorithm over three independent runs. The results are obtained with 3-exchange during the travelling salesman heuristic and with a tournament size of one.

Instance number	Best	% Gap	Worst	% Gap	Average	Average CPU time
1	524.61	0.00	524.61	0.00	524.61	5,691.36
2	560.24	0.87	560.89	0.98	560.46	19,377.07
3	819.56	0.00	824.78	0.64	821.30	24,940.50
4	866.37	0.00	866.37	0.00	866.37	34,121.87

The modification turns out to be less good for the fourth vehicle routing instance when the Pure Genetic Algorithm is used to find a near-optimal solution. An explanation for this observation can be that the vehicles in this instance also have a distance limit. When all population member have the same chance to be selected as a parent, more offspring with an unfitness value larger than zero will be generated. There was less diversity between the individuals in the end population for this instance

than for the other three. To show this, we have randomly selected one customer for both instances to represent the number of times that the selected customer is supplied by the same vehicle as the other customers, for all population members together. The results are shown in Figure 5.6 and Figure 5.7. The *n* by *n* matrix with the number of candidate solutions in which customer *i* is supplied by the same vehicle as customers *j*, where *i,j* ϵ V\{*0*}, can be requested by the author of this article. From the figures below it arises that there is more diversity between the individuals in the final population for the first vehicle routing instance than for the fourth instance.

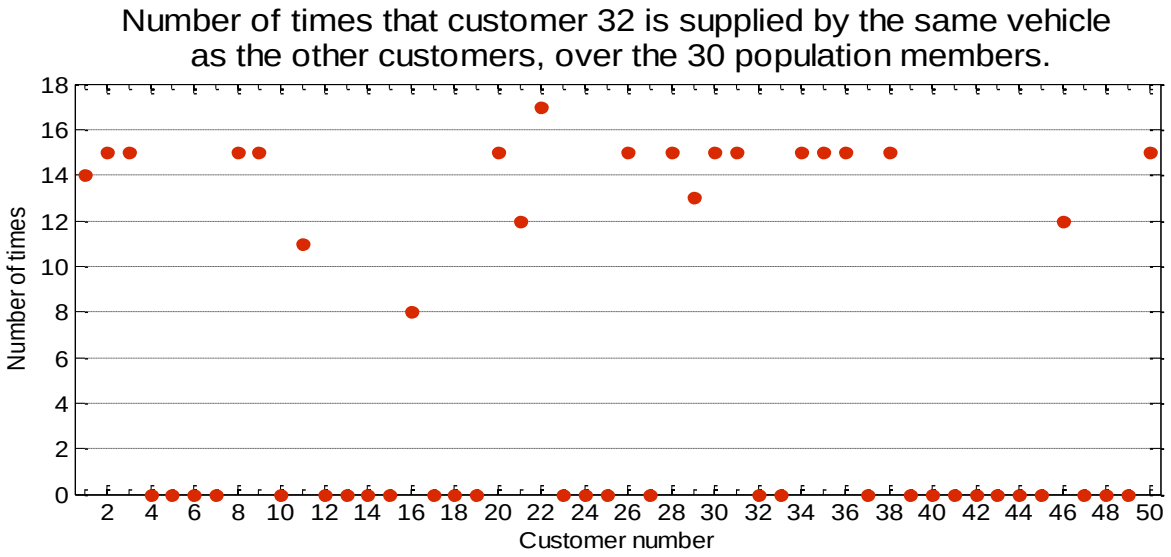


Figure 5.6: Number of times that customer 32 is supplied by the same vehicle as other customers, for all population members of the first instance together. These members are generated by using the Pure Genetic Algorithm without 3-exchange and a tournament size of one.

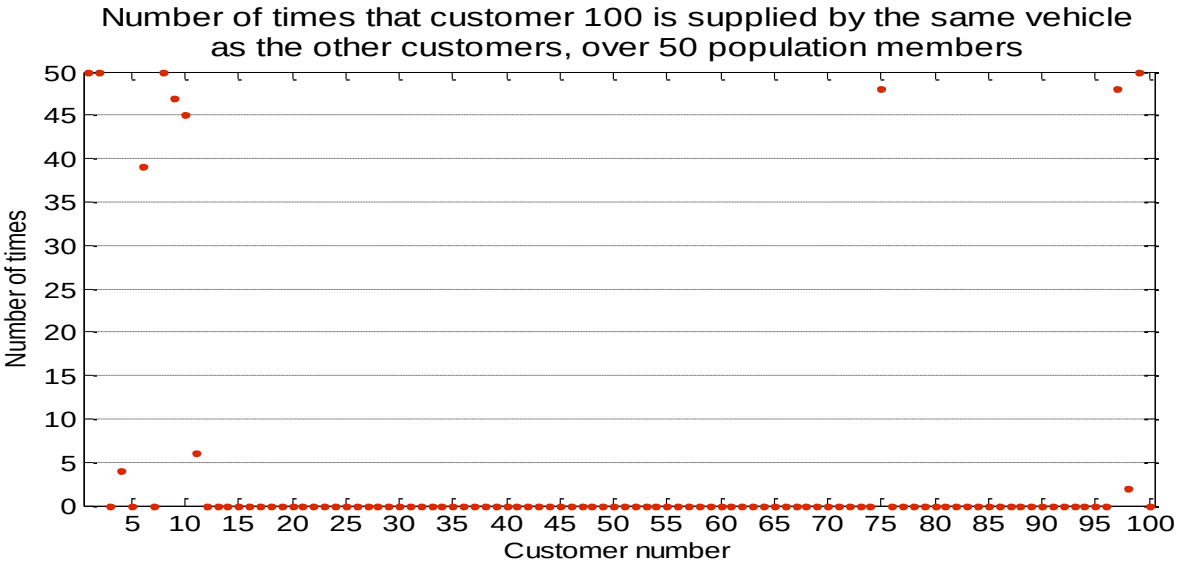


Figure 5.7: Number of times that customer 100 is supplied by the same vehicle as other customers, for all population members of the fourth instance together. These members are generated by using the Pure Genetic Algorithm without 3-exchange and a tournament size of one.

After evaluating these kind of figures for each of the customers, one may conclude that there is more variation between the individuals in the final population of the first instance than between the individuals in the final population of the fourth instance. As a result of this, it is possible that the algorithm ends in a local minimum. Further improvements are hard to attain when neighbourhood search is not incorporated.

More runs needs to be carried out to validate the values stated in Table 5.17 and Table 5.18. The obtained values seems to be pretty good in comparison with the well-known published results obtained with the tabu search implementation of Rochat and Taillard [8]. The Hybrid Genetic Algorithm with 3-optimal refinement does better on average than the Pure Genetic Algorithm without 3-optimal refinement. That coincides with our expectations, because the 3-optimal refinement may be an important tool to reduce the distance travelled by the vehicles and the movement to neighbourhood solution may avoid premature convergence to a local minimum.

The long average computing times stated in Table 5.18 are mainly due to the fact that the improvement in the fitness value of the best individual depends hardly on the initial population. For example, the worst fitness value for the third vehicle routing instance over three independent runs is found in 15,137 seconds, while the program needs more than 34,000 seconds to find a fitness value of 819.56. In the last case, the neighbourhood search must be executed six times, while it is only executed three times when a fitness value of 824.78 is found.

Table 5.19 shows the results that were obtained by carrying out three independent runs of the Hybrid Genetic Algorithm when all unique offspring are added to the population, when there is for both offspring an individual that has a higher fitness and/or unfitness value. The best offspring, the one with the lowest unfitness value with ties broken by lowest fitness, is added first.

Table 5.19: Performance of the Hybrid Genetic Algorithm over three independent runs. The results are obtained with 3-exchange during the travelling salesman heuristic and all unique offspring can be added to the population.

Instance number	Best	% Gap	Worst	% Gap	Average	Average CPU time
1	524.61	0.00	524.61	0.00	524.61	8,823.94
2	560.24	0.87	562.65	1.30	561.85	31,631.02
3	845.37	3.15	853.43	4.13	848.06	14,166.90
4	866.37	0.00	869.26	0.33	867.34	28,645.81

When we compare Table 5.19 with Table 5.18, it is clear that the results obtained with a tournament size of one have a better quality. However, the computing times differ significantly for all vehicle routing instances. The next paragraphs are used to discuss the possible causes for these differences in average computing time.

The observation that the average computing time for the first instance is longer in the case that both instances are added is caused by the fact that this results in less diversity within the population. Both algorithm stops after 20,000 successful runs, because there was no improvement found in those runs. There are more unsuccessful generations produced with the second variant, where both offspring are added, than with a tournament size of one. The average number of generation – successful and unsuccessful – is 33,129 for the first variant of the Hybrid Genetic Algorithm and 45,562 for the second variant of the Hybrid Genetic Algorithm. This can be an explanation for the difference in average computing time.

The variant where the tournament size is one performs on average better for the second instance than the variant in which both offspring are added to the population, both in quality and solution time. The difference in computing time is mainly caused by the fact that the first variant terminates after 20,350 successful generations and the second variant after 37,666 successful generations, on average. This means that the neighbourhood search method is applied more often in the second case. The number of generations – successful and unsuccessful – is also much higher for the second variant, e.g. 138,941 compared to 67,372 for the first variant.

The solution quality obtained for the third routing instance is much better when the tournament size is set equal to one than when both offspring are added to the population. The computing time depends strongly on the initial population and varies significantly between different runs carried out with the same algorithm. One run of the Hybrid Genetic Algorithm with a tournament size of one needs 50,000 number of successful generations before termination and another run terminates after 20,050 successful generations. The difference in computing time is large for independent runs carried out with the same algorithm, especially for the first variant.

The difference in computing time for the two variants of the Hybrid Genetic Algorithm carried out for the fourth instance is caused by the fact that the neighbourhood method must be performed one time extra for the first variant, on average.

As can be seen from this evaluation, the results obtained with the Hybrid Genetic Algorithm with a tournament size of one are one average better than the results obtained with the Hybrid Genetic Algorithm when all unique offspring are added to the population, in terms of solution quality. The computing times are volatile, also over independent runs carried out with the same algorithm. The convergence speed is, among other things, influenced by the diversity between the individual in the initial population and the quality of the candidate solutions in the initial population.

6. Conclusion, improvements and future research

This chapter is used to provide the reader with a conclusion about the performance of the Genetic Algorithm. Recommendations for future research are provided.

The Pure Genetic Algorithm described by Baker and Ayechew [1] performs well. Although, the fitness values of the best found individuals are, for all vehicle routing instances, somewhat higher than the ones found with the tabu search implementation of Rochat and Taillard [8]. There is also quite some volatility in the performance of the Pure Genetic Algorithm without 3-optimal refinement from one run to another for the last two vehicle routing instances. The Hybrid Genetic Algorithm performs on average better than the Pure Genetic Algorithm, but not for all instances.

For some vehicle routing instances that we have investigate, the performance of the Pure Genetic Algorithm described by Baker and Ayechew [1] can be improved. The best found individual over three independent runs with a tournament size of three and the best found individual over three independent runs when both offspring can be added to the population have lower fitness values than the best individual found with the standard Pure Genetic Algorithm for the third vehicle routing instance. All individuals found are feasible, in the sense that no constraint violation occurs.

On average, the best results were obtained by using the Hybrid Genetic Algorithm with 3-optimal refinement during the travelling salesman heuristic and a tournament size of one. In that case, the best found individual over three independent runs is for three vehicle routing instances equal to the results obtained by using tabu search [8]. The best found result for the second vehicle routing instance is 0.87 percent higher than the distance travelled in the best found solution by Rochat and Taillard [8]. When the Pure Genetic Algorithm described by Baker and Ayechew [1] is used to find a near-optimal, feasible solution for this instance, a fitness value which is 0.23 percent higher than the distance travelled in the best found solution by Rochat and Taillard [8] is found.

The excessive solution times necessary to find near-optimal solutions are probably the most troublesome outcome of this study. However, the solution times found by Baker and Ayechew [1] are not excessive and the method used to find near-optimal tours is basically the same. The excessive computing times are mainly caused by the programming environment.

6.1 Recommendations for future research

We will now provide some variations of the algorithm that can be tried to improve the performance of the routing heuristic.

First of all, the modifications that are discussed in Section 3.3 should be carried out for each of the four selected vehicle routing instances, not only for the third instance. Besides that, it is also a good idea to investigate the other ten vehicle routing instances that were discussed by Baker and Ayechew [1] and which can be downloaded from the OR-library, see Beasley [16].

The problem is not fully realistic now. In practice, there are different types of vehicles. On that account, we must adjust the heuristic in such a way that there multiple types of vehicles available, each with their own specifications. This is called the Heterogeneous Vehicle Routing Problem. After the introduction of different types of vehicles, the fixed costs associated with each vehicle must also taken into. Our aim is to minimise the total costs, under the restriction that the demand of customers is fulfilled and all restrictions hold. A vehicle with more capacity and/or a higher maximum travel distance is more expensive than a vehicle with less capacity. The adjustment makes the problem far more complex, because it has to deal with many different aspects.

Other modifications than the ones described in Section 3.3 are possible. Baker and Ayechew [1] fixed the population size on thirty for small instances with fifty or less customers and a population size of fifty for larger instances. When the population size is large, the time taken to find a good solution will be large. However, a small population size likely results in premature convergence. Both aspects are not desirable. Vishnu Raja and Murali Bhaskaran wrote an article [19] about possible improvements of the Genetic Algorithm when the population size is reduced. They found out that the best optimal solution can be found with the Genetic Algorithm when the best individuals are loaded in the initial population. A Population Reduction method is described in this paper, which can also be used in our study.

The process of selecting two parent solutions from the population to start crossover can be adapted. Baker and Ayechew [1] used the binary tournament method to select two parent solutions. We have already adapted this method by changing the tournament size. However, the performance of the Genetic Algorithm may improve when the candidates for the reproduction process are chosen in another way. One possibility is to use the Roulette-Wheel Selection. Given the fitness value of each of the population members, this method selects an individual e with fitness f_e with probability f_e / $\sum_{r=1}^{s} f_r$, where s > 0 the size of the population. The performance of this method can be bad when a candidate solution has a really large fitness value in comparison with the other individuals.

It may also be beneficial to generate new offspring by using three different parents instead of two. In that way, three offspring are created per generation. There are at least two methods that can be used to form the offspring. In both cases, you randomly select two different genes. Figure 6.1 and

Figure 6.2 show the first and second method that can be used to generate new offspring, respectively. Off course, the same method can be used when four or more parents are selected. Please note that reproduction method one is a special case of reproduction method two. All children that are generated with the first method can also be generated with the second method. It is also possible to select more than two different genes. This modification can also turns out to make the algorithm much worse, in the sense that individuals with a high unfitness or fitness value are created.

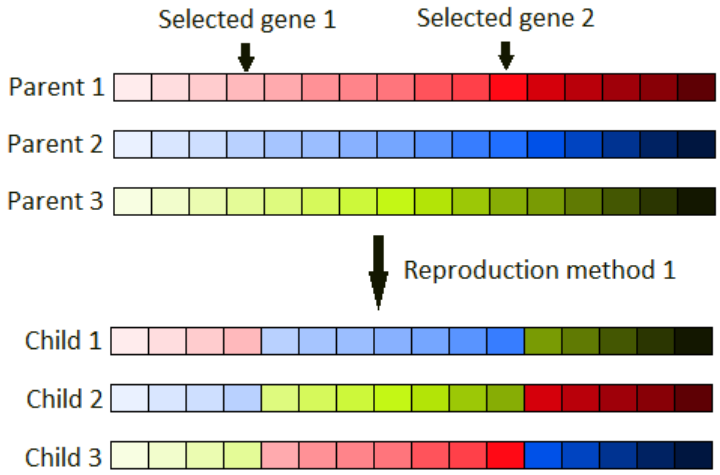


Figure 6.1: Reproduction method one when the number of selected parents is equal to three. This method is well-defined and clear.

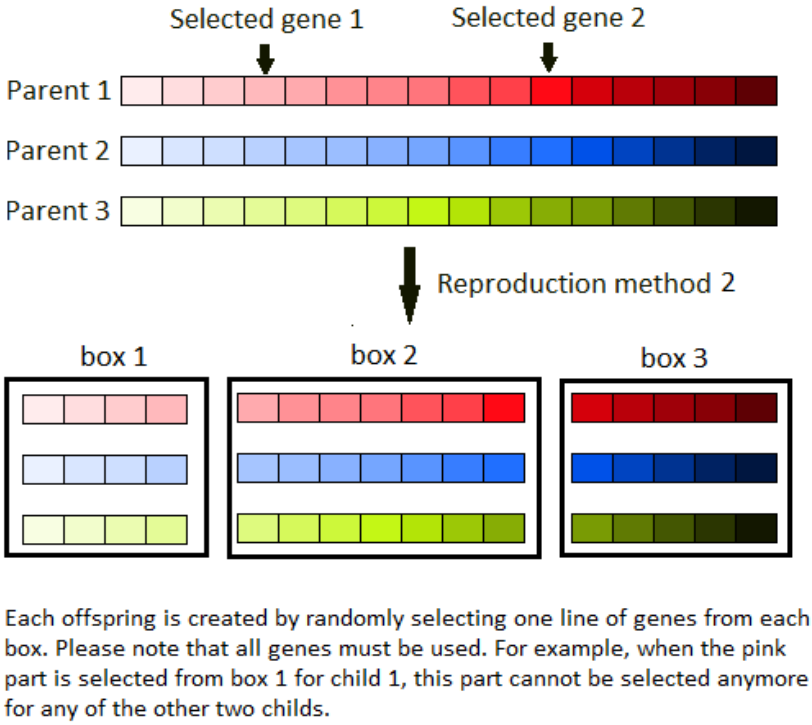


Figure 6.2: Reproduction method two when the number of selected parents is equal to three.

Other variants of the travelling salesman heuristic can be used to create the routes for each vehicle. It may be beneficial in terms of the solution quality to incorporate the V-optimal heuristic or search for randomised improvements. The V-optimal heuristic is a generalization of the k-optimal heuristic, in the sense that the number of edges that is removed is varied. This number increases when the search process continues. There are many other variations of the travelling salesman heuristic available.

There are many other mutations that can be applied to newly generated offspring. It may be beneficial to shift one of the customers allocated to the vehicle with the longest travelling distance to a neighbouring vehicle. Even when this mutations is applied, different variations are possible. On the one hand, you can put the restriction that such a shift should only be maintained when it does not increase the unfitness and/or the fitness value of the individual. Another possibility would be to maintain the shift, regardless of a possible increase in the unfitness and/or the fitness value. When the offspring duplicates another population member after the mutation, one can decide to discard the offspring or to undo the mutation.

It is also a good idea to test different combinations of modifications. These may improve the solution quality even further than individual modifications.

Bibliography

[1] Baker BM, Ayechew MA. A Genetic Algorithm for the Vehicle Routing Problem. Computers & Operations Research 2003;30:787-800.

[2] Dantzig GB, Ramser JH. The truck dispatching problem. Management Science 1959;6:80-91.

[3] Multiple Depot VRP. 2013. Retrieved May 7, 2014, from http://neo.lcc.uma.es/vrp/vrp-flavors/multiple-depot-vrp/.

[4] Dethloff J. Vehicle routing and reverse logistics: The Vehicle Routing Problem with simultaneous delivery and pick-up, OR Spectrum 2001;23:79-96.

[5] Beasley JE, Christofides N. Vehicle routing with a sparse feasibility graph. European Journal of Operational Research 1997;98:499-511.

[6] Toth P, Vigo D. Models, relaxations and exact approaches for the capacitated Vehicle Routing Problem. Discrete Applied Mathematics 2002;123:487-512.

[7] Taillard E. Parallel iterative search methods for Vehicle Routing Problems. Networks 1993;23:661-73.

[8] Rochat Y, Taillard E. Probabilistic diversification and intensification in local search for Vehicle Routing Problem. Journal of Heuristics 1995;1:147-67.

[9] Osman IH. Metastrategy simulated annealing and Tabu search algorithms for the vehicle routing problem. Operations Research 1993;41:421-51.

[10] Toth P, Vigo D. The Vehicle Routing Problem. Monographs on Discrete Mathematics and Applications 2002.

[11] Gillett BE, Miller LR. A heuristic algorithm for the vehicle dispatch problem. Operations Research 1974;22:340-9.

[12] Fisher ML, Jaikumar R. A generalised assignment heuristic for vehicle routing. European Journal of Operational Research 1999;199:147-57.

[13] Weisstein EW. Polar Angle. MathWorld, a Wolfram Web Resource.

[14] Beasley JE, Chu PC. Constraint handling in Genetic Algorithms: the set partitioning problem. Journal of Heuristics 1998;4:323–57.

[15] Vehicle Routing Problem Benchmark instances. 2014. Retrieved May 20, 2014, from http://dev.heuristiclab.com/trac/hl/core/wiki/Vehicle Routing Problem.

[16] Beasley JE. OR-Library: distributing test problems by electronic mail. Journal of the Operational Research Society 1990;41:1069-72.

[17] Ralphs T. Vehicle Routing Data Sets. Retrieved June 7, 2014, from http://www.coin-or.org/SYMPHONY/branchandcut/VRP/data/index.htm.

[18] Augerat P, Belenguer JM, Benavent E, Corberán A, Naddef D, Rinaldi G. Computational Results with a Branch and Cut Code for the Capacitated Vehicle Routing Problem. Technical Report 949-M 1995.

[19] Vishnu Raja P, Murali Bhaskaran V. Improving the Performance of Genetic Algorithm by Reducing the Population Size. International Journal of Emerging Technology and Advanced Engineering 2013;8:86-91

List of Figures

List of Tables

Appendix – Table of contents

A1: Tours of the best found individuals for the first vehicle routing instance.

Table A1.1: Tour of the best found individual for the first instance by Rochat and Taillard [8]. This data can be downloaded from the OR-library [16].

Vehicle number	Fitness value	Total demand	Route of the different vehicles in the fleet. The depot is represented by 0.												
1	98.45	152.00	0	27	48	23	7	43	24	25	14	6	0		
2	109.06	157.00	0	47	4	17	42	19	40	41	13	18	0		
3	99.25	160.00	0	46	5	49	10	39	33	45	15	44	37	12	0
4	99.33	159.00	0	38	9	30	34	50	16	21	29	2	11	0	
5	118.52	149.00	0	32	1	22	20	35	36	3	28	31	26	8	0
Total:	524.61	777.00													

Table A1.2: Tour of the best found individual associated with the lowest fitness value represented in Table 5.2 for the first vehicle routing instance, found with the Pure Genetic Algorithm.

Vehicle number	Fitness value	Total demand	Route of the different vehicles in the fleet. The depot is represented by 0.												
1	118.52	149.00	0	32	1	22	20	35	36	3	28	31	26	8	0
2	98.45	152.00	0	27	48	23	7	43	24	25	14	6	0		
3	109.06	157.00	0	18	13	41	40	19	42	17	4	47	0		
4	99.25	160.00	0	46	5	49	10	39	33	45	15	44	37	12	0
5	99.35	159.00	0	11	16	2	29	21	50	34	30	9	38	0	
Total:	524.63	777.00													

Table A1.3: Tour of the best found individual associated with the fitness value represented in Table 5.2 for the first vehicle routing instance, found with the Hybrid Genetic Algorithm.

Vehicle number	Fitness value	Total demand	Route of the different vehicles in the fleet. The depot is represented by 0.												
1	98.45	152.00	0	27	48	23	7	43	24	25	14	6	0		
2	109.06	157.00	0	47	4	17	42	19	40	41	13	18	0		
3	99.25	160.00	0	46	5	49	10	39	33	45	15	44	37	12	0
4	99.33	159.00	0	38	9	30	34	50	16	21	29	2	11	0	
5	118.52	149.00	0	32	1	22	20	35	36	3	28	31	26	8	0
Total:	524.61	777.00													

A2: Tours of the best found individuals for the second vehicle routing instance.

Table A2.1: Tour of the best found individual for the second instance by Rochat and Taillard [8]. This data can be downloaded from the OR-library [16].

Vehicle number	Fitness value	Total demand	Route of the different vehicles in the fleet. The depot is represented by 0.											
1	99.12	155.00	0	5	49	10	39	33	45	15	44	37	12	0
2	108.08	137.00	0	1	22	31	28	3	36	35	20	2	0	
3	109.94	131.00	0	17	42	19	40	41	13	25	14	0		
4	42.33	80.00	0	46	47	4	18	0						
5	100.64	133.00	0	27	48	8	26	7	43	24	23	6	0	
6	95.33	141.00	0	38	9	30	34	50	21	29	16	11	32	0
Total:	555.43	777.00												

Table A2.2: Tour of the best found individual associated with the lowest fitness value represented in Table 5.2 for the second vehicle routing instance, found with the Pure Genetic Algorithm.

Vehicle number	Fitness value	Total demand	Route of the different vehicles in the fleet. The depot is represented by 0.											
1	108.08	137.00	0	1	22	31	28	3	36	35	20	2	0	
2	100.64	133.00	0	6	23	24	43	7	26	8	48	27	0	
3	50.46	90.00	0	14	25	18	0							
4	109.07	132.00	0	17	44	42	19	40	41	13	4	47	0	
5	93.10	144.00	0	12	37	15	45	33	39	10	49	5	46	0
6	95.33	141.00	0	32	11	16	29	21	50	34	30	9	38	0
Total:	556.68	777.00												

Table A2.3: Tour of the best found individual associated with the fitness value represented in Table 5.2 for the second vehicle routing instance, found with the Hybrid Genetic Algorithm.

Vehicle number	Fitness value	Total demand	Route of the different vehicles in the fleet. The depot is represented by 0.											
1	98.76	123.00	0	22	3	36	35	20	29	2	32	0		
2	79.46	94.00	0	1	28	31	26	8	48	27	0			
3	91.63	120.00	0	14	25	24	43	7	23	6	0			
4	109.06	157.00	0	18	13	41	40	19	42	17	4	47	0	
5	99.16	155.00	0	5	49	10	39	33	45	15	44	37	12	0
6	82.88	128.00	0	46	38	9	30	34	21	50	16	11	0	
Total:	560.89	777.00												

A3: Tours of the best found individuals for the third vehicle routing instance.

Table A3.1: Tour of the best found individual for the third instance by Rochat and Taillard [8]. This data can be downloaded from the OR-library [16].

Vehicle number	Fitness value	Total demand	Route of the different vehicles in the fleet. The depot is represented by 0.															
1	43.59	150.00	0	66	62	74	63	65	67	0								
2	101.88	200.00	0	59	60	58	56	53	54	55	57	0						
3	50.80	170.00	0	21	22	23	26	28	30	29	27	25	24	20	0			
4	137.02	200.00	0	81	78	76	71	70	73	77	79	80	72	61	64	68	69	0
5	76.07	170.00	0	90	87	86	83	82	84	85	88	89	91	0				
6	95.94	190.00	0	98	96	95	94	92	93	97	100	99	0					
7	96.04	200.00	0	13	17	18	19	15	16	14	12	10	0					
8	56.17	170.00	0	5	3	7	8	11	9	6	4	2	1	75	0			
9	97.23	200.00	0	34	36	39	38	37	35	31	33	32	0					
10	64.81	160.00	0	47	49	52	50	51	48	45	46	44	40	41	42	43	0	
Total:	819.56	1810.00																

Table A3.2: Tour of the best found individual associated with the lowest fitness value represented in Table 5.2 for the third vehicle routing instance, found with the Pure Genetic Algorithm.

Vehicle number	Fitness value	Total demand	Route of the different vehicles in the fleet. The depot is represented by 0.															
1	76.07	170.00	0	90	87	86	83	82	84	85	88	89	91	0				
2	95.94	190.00	0	99	100	97	93	92	94	95	96	98	0					
3	56.17	170.00	0	75	1	2	4	6	9	11	8	7	3	5	0			
4	96.04	200.00	0	13	17	18	19	15	16	14	12	10	0					
5	50.80	170.00	0	20	24	25	27	29	30	28	26	23	22	21	0			
6	97.23	200.00	0	32	33	31	35	37	38	39	36	34	0	0	0			
7	64.81	160.00	0	47	49	52	50	51	48	46	45	44	40	41	42	43	0	
8	101.88	200.00	0	57	55	54	53	56	58	60	59	0						
9	57.79	150.00	0	69	66	68	64	61	72	74	62	65	67	0				
10	128.04	200.00	0	81	78	76	71	70	73	77	79	80	63	0				
Total:	824.78	1810.00																

Table A3.3: Tour of the best found individual associated with the fitness value represented in Table 5.2 for the third vehicle routing instance, found with the Hybrid Genetic Algorithm.

Vehicle number	**Fitness value**	**Total demand**	**Route of the different vehicles in the fleet. The depot is represented by 0.**														
1	63.80	120.00	0	91	89	88	85	86	87	90	0						
2	95.94	190.00	0	99	100	97	93	92	94	95	96	98	0				
3	56.17	170.00	0	75	1	2	4	6	9	11	8	7	3	5	0		
4	96.04	200.00	0	13	17	18	19	15	16	14	12	10	0				
5	50.80	170.00	0	21	22	23	26	28	30	29	27	25	24	20	0		
6	97.23	200.00	0	32	33	31	35	37	38	39	36	34	0				
7	64.81	160.00	0	43	42	41	40	44	45	46	48	51	50	52	49	47	0
8	101.88	200.00	0	59	60	58	56	53	54	55	57	0					
9	59.40	200.00	0	69	66	68	64	61	72	74	62	63	65	67	0		
10	139.88	200.00	0	80	79	77	73	70	71	76	78	81	82	83	84	0	
Total:	825.95	1810.00															

A4: Tours of the best found individuals for the fourth vehicle routing instance.

Table A4.1: Tour of the best found individual for the fourth instance by Rochat and Taillard [8]. This data can be downloaded from the OR-library [16].

Vehicle number	Fitness value	Total demand	Route of the different vehicles in the fleet. The depot is represented by 0.											
1	57.79	150.00	0	67	65	62	74	72	61	64	68	66	69	0
2	101.88	200.00	0	57	59	60	58	56	53	54	55	0		
3	61.56	110.00	0	20	49	52	50	51	48	45	46	47	0	
4	128.04	200.00	0	63	80	79	77	73	70	71	76	78	81	0
5	49.41	160.00	0	21	22	24	25	27	29	30	28	26	23	0
6	96.70	200.00	0	1	99	100	97	93	92	94	95	96	98	0
7	96.04	200.00	0	13	17	18	19	15	16	14	12	10	0	
8	45.47	60.00	0	43	42	44	40	41	0					
9	76.07	170.00	0	90	87	86	83	82	84	85	88	89	91	0
10	97.23	200.00	0	32	33	31	35	37	38	39	36	34	0	
11	56.17	160.00	0	75	2	4	6	9	11	8	7	3	5	0
Total:	866.37	1810.00												

Table A4.2: Tour of the best found individual associated with the lowest fitness value represented in Table 5.2 for the fourth vehicle routing instance, found with the Pure Genetic Algorithm.

Vehicle number	Fitness value	Total demand	Route of the different vehicles in the fleet. The depot is represented by 0.											
1	76.07	170.00	0	90	87	86	83	82	84	85	88	89	91	0
2	96.70	200.00	0	1	99	100	97	93	92	94	95	96	98	0
3	56.17	160.00	0	75	2	4	6	9	11	8	7	3	5	0
4	96.04	200.00	0	13	17	18	19	15	16	14	12	10	0	
5	49.41	160.00	0	21	22	24	25	27	29	30	28	26	23	0
6	97.23	200.00	0	32	33	31	35	37	38	39	36	34	0	
7	63.08	120.00	0	43	47	46	45	48	51	50	52	49	20	0
8	102.22	200.00	0	42	44	60	58	56	53	54	55	68	69	0
9	87.53	180.00	0	41	40	59	57	64	61	72	74	62	66	0
10	29.35	70.00	0	67	65	63	0							
11	127.30	150.00	0	81	78	76	71	70	73	77	79	80	0	
Total:	881.11	1810.00												

Table A4.3: Tour of the best found individual associated with the fitness value represented in Table 5.2 for the fourth vehicle routing instance, found with the Hybrid Genetic Algorithm.

Vehicle number	**Fitness value**	**Total demand**	**Route of the different vehicles in the fleet. The depot is represented by 0.**											
1	76.07	170.00	0	90	87	86	83	82	84	85	88	89	91	0
2	96.70	200.00	0	1	99	100	97	93	92	94	95	96	98	0
3	56.17	160.00	0	75	2	4	6	9	11	8	7	3	5	0
4	96.04	200.00	0	13	17	18	19	15	16	14	12	10	0	
5	49.41	160.00	0	21	22	24	25	27	29	30	28	26	23	0
6	97.23	200.00	0	34	36	39	38	37	35	31	33	32	0	
7	61.56	110.00	0	47	46	45	48	51	50	52	49	20	0	
8	45.47	60.00	0	41	40	44	42	43	0					
9	101.88	200.00	0	59	60	58	56	53	54	55	57	0		
10	57.79	150.00	0	69	66	68	64	61	72	74	62	65	67	0
11	128.04	200.00	0	81	78	76	71	70	73	77	79	80	63	0
Total:	866.37	1810.00												

B1: Tours of the best found individuals for instance A-n32-k5

Table B1.1: Optimal tour found by Augerat et al. [18] for instance A-n32-A5.

Vehicle number	Fitness value	Total demand	Route of the different vehicles in the fleet. The depot is represented by 0.											
1	156.28	98.00	0	21	31	19	17	13	7	26	0			
2	73.49	72.00	0	12	1	16	30	0						
3	59.26	44.00	0	27	24	0								
4	268.96	98.00	0	29	18	8	9	22	15	10	25	5	20	0
5	229.82	98.00	0	14	28	11	4	23	3	2	6	0		
Total:	787.81	410.00												

Table B1.2: Tour of the best found individual associated with the lowest fitness value represented in Table 5.9 for instance A-n32-k5, found with the Pure Genetic Algorithm.

Vehicle number	Fitness value	Total demand	Route of the different vehicles in the fleet. The depot is represented by 0.											
1	268.84	98.00	0	18	8	9	22	15	29	10	25	5	20	0
2	59.26	44.00	0	24	27	0								
3	231.72	100.00	0	26	6	3	2	23	4	11	28	14	0	
4	155.75	96.00	0	7	13	17	19	31	21	0				
5	73.49	72.00	0	30	16	1	12	0						
Total:	789.06	410.00												

Table B1.3: Tour of the best found individual associated with the lowest fitness value represented in Table 5.9 for instance A-n32-k5, found with the Hybrid Genetic Algorithm.

Vehicle number	Fitness value	Total demand	Route of the different vehicles in the fleet. The depot is represented by 0.											
1	268.84	98.00	0	18	8	9	22	15	29	10	25	5	20	0
2	59.26	44.00	0	24	27	0								
3	229.21	98.00	0	6	3	2	23	4	11	28	14	0		
4	156.28	98.00	0	21	31	19	17	13	7	26	0			
5	73.49	72.00	0	30	16	1	12	0						
Total:	787.08	410.00												

B2: Tours of the best found individuals for instance A-n48-k7.

Table B2.1: Optimal tour found by Augerat et al. [18] for instance A-n48-k7.

Vehicle number	Fitness value	Total demand	Route of the different vehicles in the fleet. The depot is represented by 0.										
1	205.92	99.00	0	34	9	24	4	11	42	15	27	45	0
2	171.99	98.00	0	32	36	38	19	25	22	6	0		
3	206.18	99.00	0	40	7	8	39	26	20	3	37	0	
4	138.90	96.00	0	41	2	10	47	17	14	0			
5	39.27	43.00	0	44	35	18	0						
6	202.68	100.00	0	28	29	21	30	13	46	33	16	0	
7	109.39	91.00	0	23	43	31	1	5	12	0			
Total:	1074.34	626.00											

Table B2.2: Tour of the best found individual associated with the lowest fitness value represented in Table 5.9 for instance A-n48-k7, found with the Pure Genetic Algorithm.

Vehicle number	Fitness value	Total demand	Route of the different vehicles in the fleet. The depot is represented by 0.										
1	138.90	96.00	0	41	2	10	47	17	14	0			
2	202.68	100.00	0	16	33	46	13	30	21	29	28	0	
3	205.92	99.00	0	34	9	24	4	11	42	15	27	45	0
4	206.18	99.00	0	37	3	20	26	39	8	7	40	0	
5	171.99	98.00	0	6	22	25	19	38	36	32	0		
6	39.27	43.00	0	18	35	44	0						
7	109.39	91.00	0	23	43	31	1	5	12	0			
Total:	1074.34	626.00											

Table B2.3: Tour of the best found individual associated with the lowest fitness value represented in Table 5.9 for instance A-n48-k7, found with the Hybrid Genetic Algorithm.

Vehicle number	Fitness value	Total demand	Route of the different vehicles in the fleet. The depot is represented by 0.										
1	138.90	96.00	0	41	2	10	47	17	14	0			
2	202.68	100.00	0	16	33	46	13	30	21	29	28	0	
3	205.92	99.00	0	34	9	24	4	11	42	15	27	45	0
4	206.18	99.00	0	37	3	20	26	39	8	7	40	0	
5	171.99	98.00	0	6	22	25	19	38	36	32	0		
6	39.27	43.00	0	18	35	44	0						
7	109.39	91.00	0	23	43	31	1	5	12	0			
Total:	1074.34	626.00											